ITEM NO. 22 MAY 28, 1982
NEW VERSIONS OF )FNS, )VARS, )GRPS, ⎕WS AND ⎕NL NOW RUNNING ON INTERN.

INTERN IS NOW RUNNING NEW VERSIONS OF )FNS, )VARS, )GRPS, ⎕WS AND ⎕NL. THE USER EVIDENT CHANGES ARE AS FOLLOWS:

1) )FNS, )VARS, AND )GRPS RUN IMMENSELY FASTER.
2) THE OVERALL PERFORMANCE OF ⎕WS IS IMPROVED, THOUGH SOME SPECIFIC ⎕WS FUNCTIONS RUN SLIGHTLY SLOWER. MORE IMPORTANTLY, ALL ERROR CHECKING (EXCEPT FOR WS FULL) IS NOW DONE BEFORE ANY ATTEMPT IS MADE TO BUILD A RESULT. THUS IT IS NO LONGER POSSIBLE TO HAVE DOMAIN ERRORS MASKED BY WS FULL.
3) ⎕NL IS NOW SORTED ALPHABETICALLY.

ITEM NO. 19 MAR 19, 1982
NEW WORKSPACE DOCUMENTATION FACILITY NOW AVAILABLE

A NEW VERSION OF THE WORKSPACE DOCUMENTATION FACILITY (WSDOC) IS NOW AVAILABLE. THE NEW PACKAGE REPLACES <7 WSDOC>, AND RESIDES IN THREE WORKSPACES: <7 WSTOFILE>, <7 STATE>, AND <7 WSDOC>. THE PREVIOUS WSDOC PACKAGE IS STILL AVAILABLE IN WORKSPACE <499 WSDOC7>.

THE NEW WSDOC PACKAGE ENABLES USERS TO OBTAIN LISTINGS AND STATIC ANALYSES OF THE ENTIRE CONTENTS OF AN APL WORKSPACE. REPORTS ARE AVAILABLE TO DISPLAY FUNCTIONS, VARIABLES (INCLUDING ARRAYS AND PACKAGES), AND GROUPS, IN VARYING LEVELS OF DETAIL. IN ADDITION, OPTIONS ARE PROVIDED TO FORMAT AND PRINT SEVERAL TYPES OF CROSS-REFERENCE REPORTS, WHICH LIST THE USE OF IDENTIFIERS WITHIN THE WORKSPACE, AND TREE STRUCTURE DIAGRAMS, WHICH LIST THE HIERARCHICAL CALLING SEQUENCE OF PROGRAMS.

THE USER IS PROVIDED WITH COMPLETE CONTROL OVER THE DETAIL AND COMPOSITION OF EACH WSDOC REPORT. IN ADDITION, OUTPUT MAY BE PRINTED LOCALLY AT THE TERMINAL OR SPOOLED TO FILES, AND MAY BE OPTIONALLY HIGH-SPEED PRINTED AT THE SHARP APL COMPUTER CENTRE.

INFORMATION ON THE USE OF THE NEW WSDOC FACILITY MAY BE OBTAINED BY DISPLAYING <DESCRIBE> AND <WSDOCHOW> IN WORKSPACE <7 STATE>.

ITEM NO. 18 MAR 12, 1982
EXTENSIONS TO EVENTS AND EVENT TRAPPING

THIS NEWS ITEM OUTLINES A NUMBER OF CHANGES RELATING TO EVENTS AND EVENT TRAPPING IN SHARP APL. UNLESS OTHERWISE NOTED, THE CHANGES ARE EFFECTIVE IMMEDIATELY.

- THE EXPERIMENTAL EVENT 2002 HAS BEEN SUPERSEDED BY ACTION CODE <I>, AND IS NOW OBSOLETE. EFFECTIVE 27 MARCH 1982, THIS EVENT WILL BE DESUPPORTED. APPLICATIONS WHICH MAKE USE OF IT SHOULD BE CONVERTED IMMEDIATELY.

- PTRDR IS NO LONGER RESET IF AN ERROR OCCURS AND IS TRAPPED.

- BLANK AND NUMERIC DIGITS ARE NO LONGER VALID DELIMITERS IN THE VECTOR FORM OF ⎕TRAP .

- ACTION CODE <D> HAS BEEN EXTENDED TO EVENT CLASSES 0 AND 1000. THUS, TO CAUSE THE ACTIVE WORKSPACE TO BE CLEARED WHEN ANY ERROR OCCURS, THE

FOLLOWING TRAP EXPRESSION MAY BE USED:

```
      ⎕TRAP←'o 0 D CLEAR'
```

WHEN THE KEYWORD 'EXIT' IS USED, THE SYSTEM REACTS BY PROPAGATING THE EVENT (WHICH MAY BE AN INTERRUPT OF SOME SORT) UP ONE STACK LEVEL AND SIGNALLING THE EVENT THERE. THE EVENT MAY BE TRAPPED BY ANOTHER TRAP EXPRESSION AT THIS POINT, OR POSSIBLY BY THE SAME <D> TRAP (IN WHICH CASE THE CUTBACK AND SIGNALLING PROCEDURE IS REPEATED). FOR EXAMPLE:

```
      ∇ FN;⎕TRAP
[1]     ⎕TRAP←'o 0 1000 D EXIT'
[2]     * A INTENTIONAL ERROR
      ∇

      FN
SYNTAX ERROR
      FN
      ^

      )SI

      S∆FN←1
      FN

      )SI

      ⎕ER
1001 STOP
      FN
      ^
```

- ACTION CODES <C>, <E>, <D>, <N>, AND <S> MAY NOW BE USED WITHOUT ANY EVENT NUMBERS. WHEN EVENT NUMBERS ARE ELIDED FOR A PARTICULAR ACTION CODE, THE CONCOMITANT TRAP EXPRESSION WILL TRAP ALL EVENTS WITHIN THE DOMAIN OF THE ACTION CODE. FOR EXAMPLE, ACTION CODES <C> AND <E>, WHEN USED WITHOUT EVENT NUMBERS, WILL TRAP ALL CLASS 0 AND 1000 EVENTS; ACTION CODE <D> WILL TRAP CLASS 0 AND 1000 EVENTS, AS WELL AS 2001; ACTION CODES <N> AND <S> WILL TRAP ALL CLASS 0, 1000, AND 2000 EVENTS.

  THE EXACT SERIES OF EVENTS WHICH IS TRAPPABLE VIA A PARTICULAR ACTION CODE MAY CHANGE AS NEW EVENTS ARE INTRODUCED INTO THE SYSTEM. FOR EXAMPLE, WHILE 'o 0 1000 2000 S' IS EQUIVALENT TO 'o S' AT PRESENT, THE INTRODUCTION OF A NEW CLASS OF EVENTS (SUCH AS 3000) MAKES THE FORMER STATEMENT WEAKER THAN THE LATTER. THEREFORE, TO PREVENT THE TRAPPING OF ANY EXISTING OR FUTURE EVENTS BY ANOTHER ⎕TRAP STATEMENT, ACTION CODE <S> WITH ELIDED NUMBERS SHOULD ALWAYS BE USED.

- A NEW ACTION CODE, <O> (OWN), HAS BEEN INTRODUCED. ACTION CODE <O> IS THE FIRST ACTION CODE WHICH BEHAVES AS A QUALIFIER, AND IS USED IN CONJUNCTION WITH ANOTHER ACTION CODE. <O> SPECIFIES THAT THE ASSOCIATED TRAP DEFINITION IS AN 'OWN TRAP' OF THE STACK LEVEL AT WHICH ⎕TRAP IS BOUND. ONLY EVENTS OCCURRING AT THAT STACK LEVEL, AND NOT THOSE OCCURRING BELOW IT, ARE TRAPPABLE BY SUCH TRAP EXPRESSIONS.

  THE <O> QUALIFIER MAKES IT POSSIBLE FOR A FUNCTION TO DISABLE ITS TRAPS FOR FUNCTIONS IT CALLS, WITHOUT RELYING ON THE CALLED FUNCTIONS THEMSELVES TO DO IT. FOR EXAMPLE:

```
      ⎕TRAP←'o 0 O E RECOVER'
```

IF THIS VALUE IS SET GLOBALLY, THEN IT WILL TRAP ONLY GLOBAL ERRORS. NOTE THAT THE OWN QUALIFIER APPLIES ONLY TO THE TRAP DEFINITION OF WHICH IT IS A PART, AND NOT TO ANY WHICH MAY PRECEDE OR FOLLOW IT.

- EXPLICIT RANGES ARE NOW PERMISSIBLE IN ⎕TRAP EVENT NUMBER SPECIFICATIONS. FOR EXAMPLE, THE RANGE 1-16 WILL TRAP ALL EVENTS IN THE RANGE 1 THROUGH 16 INCLUSIVE, WHICH REPRESENTS THE COMMON APL ERRORS. FORMALLY, CLASS 0 IS EQUIVALENT TO THE RANGE 1-999, AND CLASS 1000 IS EQUIVALENT TO THE RANGE 1001-1999 . RANGES ARE PARTICULARLY VALUABLE WHEN SPECIFYING CLASSES OF USER-DEFINED ERRORS. FOR EXAMPLE:

```
⎕TRAP←'○ 1-499 600-699 1000 C SAVE'
```

- EFFECTIVE 27 MARCH 1982, THE LOCAL DEFAULT OF ⎕TRAP WILL BE CHANGED FROM '' TO 2 6 ⍴'D EXITS ' . THE GLOBAL DEFAULT OF ⎕TRAP REMAINS AN EMPTY VECTOR.

THE NEW LOCAL VALUE PROVIDES A FAIL-SAFE DEFAULT, DESIGNED TO GREATLY SIMPLIFY THE WRITING OF SECURE PROGRAMS IN SHARP APL. THE EFFECT OF THE NEW DEFAULT IS TO PROPAGATE ALL EVENTS UP THE STACK ONE LEVEL, BY CUTTING BACK THE STACK AND SIGNALLING THE EVENT TO THE CALLER. THIS PROCESS CONTINUES UNTIL 'D EXIT' IS NOT THE MOST LOCAL TRAP DEFINITION.

THE NEW LOCAL DEFAULT OF ⎕TRAP PROVIDES PROTECTION FACILITIES ANALOGOUS TO THOSE DEMONSTRATED BY APL PRIMITIVE FUNCTIONS. THAT IS, BY SIMPLY LOCALIZING ⎕TRAP AND NOT DEFINING IT ANYWHERE WITHIN THE BODY OF THE PROGRAM, THE PROGRAM WILL BEHAVE SIMILAR TO THE WAY PRIMITIVE FUNCTIONS DO: EXECUTION OF THE FUNCTION WILL BE TERMINATED BY ANY ERROR OCCURRING WITHIN IT, OR BY ANY ASYNCHRONOUS EVENT OCCURRING WHILE THE FUNCTION IS IN EXECUTION. IN SHORT, THE FUNCTION IS NEVER LEFT SUSPENDED ON THE EXECUTION STACK.

THE NEW BEHAVIOUR MAKES IT POSSIBLE FOR A PROGRAM AUTHOR TO ENSURE THAT A PROGRAM HAS A CHANCE TO CONDITION ITS ENVIRONMENT (SET EVENT TRAPS, ORIGIN, AND SO ON) BEFORE ANY INTERRUPTION CAN OCCUR. THIS CAN AFFECT EXISTING PROGRAMS WHICH LOCALIZE ⎕TRAP BUT DO NOT DEFINE IT UNTIL SOME TIME LATER. IN PARTICULAR, AN ERROR OCCURRING BEFORE ⎕TRAP IS RESET BY THE PROGRAM WOULD NOW CAUSE EXECUTION TO BE ABANDONED. THIS BRHAVIOUR IS EASILY REMEDIED BY AN EARLY ASSIGNMENT OF ⎕TRAP, IN CASES WHERE IT IS IMPORTANT.

ITEM NO. 17 FEB 19, 1982

## IMPROVED DISPLAY FOR ARRAYS (AND A NEW SYSTEM VARIABLE, ⎕PS)

MONADIC ⍕ (THORN) AND THE SYSTEM DEFAULT DISPLAY (⎕←) NOW FORMAT ENCLOSED ARRAYS. THE DEFINITION IS BASED ON A PROPOSAL BY JENKINS AND MICHEL [1]. IT HAS BEEN EXTENDED TO INCLUDE AN OPTION TO DRAW BOXES AROUND ENCLOSED SUBARRAYS, AS USED BY GHANDOUR AND MEZEI [2].

⍕ CONVERTS EACH SUBARRAY TO A CHARACTER REPRESENTATION. THE CHARACTER ARRAYS CREATED AT EACH LEVEL ARE CATENATED TO FORM A RECTANGULAR ARRAY OF RANK 2 OR GREATER, PRESERVING ALL BUT THE LAST TWO ELEMENTS OF THE SHAPE VECTOR OF THE SUBARRAY, I.E., $(-2\downarrow\rho\text{⍕}\omega) \equiv {}^{-}2\downarrow\rho\omega$ .

THE SYSTEM VARIABLE ⎕PS CONTROLS THE POSITIONING AND SPACING OF ENCLOSED ARRAYS

FOR BOTH DEFAULT DISPLAY AND MONADIC ⍕.

⎕PS (POSITIONING AND SPACING):

⎕PS IS A FOUR-ELEMENT VECTOR, THE FIRST TWO ELEMENTS CONTROL POSITIONING (JUSTIFICATION) OF THE SUBARRAYS, THE LAST TWO CONTROL SPACING.

1. POSITION

EACH SUBARRAY WILL BE PLACED IN ONE OF NINE POSITIONS WITHIN ITS 'PRINT BOX', I.E.,

| | | |
|---|---|---|
| TOP,LEFT | TOP,CENTRE | TOP,RIGHT |
| CENTRE,LEFT | CENTRE,CENTRE | CENTRE,RIGHT |
| BOTTOM,LEFT | BOTTOM,CENTRE | BOTTOM,RIGHT |

2. SPACING

YOU MAY INSERT ANY NUMBER OF SPACES BETWEEN THE SUBARRAYS OF THE RESULT. BOXES SHOWING THE STRUCTURE MAY BE DRAWN AROUND SUBARRAYS. IF THEY ARE DRAWN, THE BOXES ENCLOSE NON-SIMPLE SCALARS, AND THE LINES ARE DRAWN IN THE SPACE BETWEEN SUBARRAYS. THE VALUE OF ⎕PS[2 3] MUST BE ≤ ¯2 IF BOXES ARE TO BE DRAWN.

| POSITIONING | | SPACING | |
|---|---|---|---|
| ⎕PS[0] | ⎕PS[1] | ⎕PS[2] | ⎕PS[3] |
| TOP ¯1 | LEFT ¯1 | NO. OF BLK ROWS BETWEEN SUBARRAYS | NO. OF BLK COLUMNS BETWEEN SUBARRAYS |
| CENTRE 0 | CENTRE 0 | | |
| BOTTOM 1 | RIGHT 1 | | |

IF ≤ ¯2
BOXES ARE DRAWN

FOR EXAMPLE:

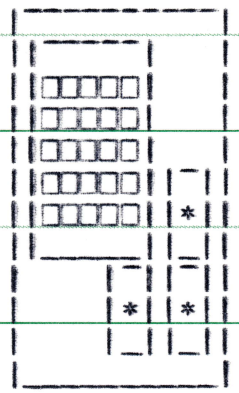
```
      A←<2 2⍴(<5 5⍴'□'),3⍴<'*'
      ⎕PS←¯1 ¯1 0 1        ⍝ DEFAULT ⎕PS
      A
┌─────┐ *                  ⍝ NOTE:   TOP LEFT POSITION
│     │                    ⍝         NO BLANK ROWS INSERTED
│     │                    ⍝         ONE BLANK COLUMN INSERTED
│     │
│     │
└─────┘
*     *

      ⎕PS←0
      A
┌─────┐                    ⍝ NOTE:   CENTERED ROWS AND COLUMNS
│     │                    ⍝         NO BLANK ROWS OR COLUMNS
│     │*
└─────┘
```

```
□□□□□
  *  *

      □PS←1 1 ¯2 ¯2

      A
```



```
⍝ NOTE:   LOWER RIGHT POSITION
          NO BLANK ROWS OR COLUMNS
          BOXES ARE DRAWN IN THE TWO SPACES
                BETWEEN ARRAYS
```

MODEL OF ENCLOSED ARRAY THORN

THE FOLLOWING MODEL OF ⍷ USES THE VARIABLE QPS FOR □PS.

```
Z←THORN D;R;C;A;□IO;W;F;E;I;J;K;G;B;T;NPS;R;B1;BXS
NPS←4⍴□IO←1
→(0=□NC 'QPS')⍴P0
NPS←4⍴QPS
P0:BXS←¯2≥¯2↑NPS
NPS[3 4]←|NPS[3 4]+2×BXS
→(A≡>A←1↑,D)/LS
I←''⍴⍴F←''
A←⍴D←((-2⌈⍴⍴D)↑ 1 1 ,⍴D)⍴D
A←(×/W←¯2↓A),¯2↑A
D←,D
L3:→((I←I+1)>⍴D)⍴L3Z
B←((-2⌈⍴⍴B)↑ 1 1 ,⍴B)⍴B←THORN>D[I]
E←((⍳¯1↑T)∈T←+\1+¯1↓0,+⌿0=T∘.|⍳J⍲((J←¯1↑T←×\1↓T),1↑T←⌽⍴B)⍴B
E←(B1⍴'¯'),[1] E,[1](B1←BXS[1],¯1↑⍴E)⍴'_'
E←(B1⍴'|'),E,(B1←(1↑⍴E),BXS[2])⍴'|'
F←F,<E
→L3
L3Z:Z←(J←0)⍴D←((B←×/2↑A),¯1↑A)⍴F
C←NPS[4]+0⌈⌈⌿≠⌈A⍴(F←⍴⍷>F)[;2]
R←NPS[3]+0⌈⌈/⌈⌿≠A⍴F[;1]
L1:→(B<J←J+1)⍴L2Z
G←((K←R[1+A[2]|¯1+J]),I←0)⍴''
L0:→(A[3]<I←I+1)⍴L0Z
E←>D[J;I]
G←G,(K,C[I])↑(-(⍴E)+⌊((0≤NPS[1 2])×(K,C[I])-NPS[3 4]+⍴E)÷2*0=NPS[1 2])↑E
→L0
L0Z:Z←Z,,G
→L1
L2Z:Z←(W,(+/R),+/C)⍴Z
Z←((-⍴⍴Z)↑-¯2↑NPS)↓Z
→0
LS:Z←⍷D
```

## FOLDING FUNCTION FOR CUT AND PASTE DISPLAY

OUTPUT OFTEN EXCEEDS ⎕PW IN WIDTH. L.M. BREED HAS PROPOSED A MATRIX ORIENTED FOLDING SCHEME SUITABLE FOR 'CUT AND PASTE' ASSEMBLY. THE FOLLOWING FUNCTION PROVIDES THIS FOLDED DISPLAY:

```
E←FOLD B;T;J
B←((-2⌈ρρB)↑ 1 1 ,ρB)ρB
E←(((⍳¯1↑T)∈T←+\1+¯1↓0,+≠0=T∘.|⍳J)⌿((J←¯1↑T←×\1↓T),1↑T←⌽ρB)ρB
E←(⎕IO+ 1 0 2)⍉((1↑ρE),T,⎕PW)⍴((1↑ρE),⎕PW×T←⌈(¯1↑ρE)÷⎕PW)↑E
```


[1] M.A. JENKINS, QUEEN'S UNIVERSITY, KINGSTON, ONTARIO, AND J. MICHEL, UNIVERSITE PARIS XI (ORSAY), 'OPERATORS IN APL WITH NESTED ARRAYS: A SYSTEM 1 VIEW'

[2] Z. GHANDOUR AND J. MEZEI, 'GENERAL ARRAYS, OPERATORS, AND FUNCTIONS', IBM JOURNAL OF RESEARCH AND DEVELOPMENT, JULY 1973.

[3] L.M. BREED, PERSONAL COMMUNICATION.


FOR MORE INFORMATION AND EXAMPLES ON ENCLOSED ARRAYS, SEE WORKSPACE 1 SATN41.

ITEM NO. 16 FEB 09, 1982

IBM3081 PROCESSOR INSTALLATION: INTERNAL SYSTEM DOWN FEB 16-18.

THE IBM 3081 PROCESSOR WILL ARRIVE THIS WEEKEND, AND WILL BE POWERED UP ON TUESDAY, 16TH FEBRUARY AT 1700 UTC (NOON TORONTO TIME). IN ORDER TO PROVIDE POWER TO THE 3081 WE MUST POWER DOWN THE AMDAHL V8 ON WHICH THE INTERNAL SYSTEM RUNS. THUS THERE WILL BE A SHUTDOWN OF THE INTERNAL SYSTEM AT 1700 UTC ON TUESDAY FEBRUARY 16TH. THE V8 WILL BE UNAVAILABLE AFTER THAT TIME.

WE EXPECT THE 3081 TO BE TURNED OVER TO US ON THURSDAY EVENING. WE WILL ATTEMPT TO MAKE THE INTERNAL SYSTEM AVAILABLE ON THAT SYSTEM AS SOON AS IT IS TURNED OVER TO US. IN THE INTERIM, WE WILL ENDEAVOUR TO PROVIDE LIMITED INTERNAL SERVICE AVAILABLE THROUGH THE /158, BUT PLEASE SCHEDULE AS LITTLE AS POSSIBLE FOR NEXT WEEK TUESDAY THROUGH FRIDAY (FEBRUARY 16-19).

ITEM NO. 14 JAN 29, 1982

NEW FUNCTION <PRTINPUT> IN WORKSPACE 1 HSPRINT

FN PRTINPUT IN 1 HSPRINT WILL NOW PERMIT PRINTING OF CONTROL MESSAGES 19 THROUGH 28 (⎕ER AND TERMINAL INPUT CONTROL MESSAGES) AS FOLLOWS:

1 PRTINPUT TIENO WILL CAUSE SUBSEQUENT INPUT AND ⎕ER CONTROL MESSAGES TO BE PRINTED. WITH A COUPLE OF EXCEPTIONS THE TEXT PRINTED WILL DUPLICATE THAT WHICH APPEARS ON THE TERMINAL DURING THE ORIGINAL SESSION CREATING THE ⎕OUT FILE.

0 PRTINPUT TIENO WILL CAUSE SUBSEQUENT INPUT AND ⎕ER CONTROL MESSAGES TO BE IGNORED.

ITEM NO. 13 JAN 29, 1982

CHANGES TO ERRORS RELATING TO NAMES AND VALUES

THE ERROR MESSAGE GENERATED WHEN A REFERENCE IS MADE TO AN OBJECT WHICH HAS NO VALUE IS NOW ALWAYS 'VALUE ERROR' (EVENT 6). PREVIOUSLY, THE MESSAGE PRODUCED WAS EITHER 'VALUE ERROR' (IF THE NAME WAS USED IN THE CONTEXT OF A NILADIC DEFINED FUNCTION), OR 'SYNTAX ERROR' (IF THE NAME WAS USED IN THE CONTEXT OF A MONADIC OR DYADIC DEFINED FUNCTION).

'VALUE ERROR' NOW MEANS THAT THE SPECIFIED NAME IS NOT BOUND TO ANY OBJECT. THIS CHANGE RATIONALIZES ERRORS RELATING TO NAMES, BY PROVIDING A SINGLE MESSAGE GOVERNING A REFERENCE TO AN UNDEFINED OBJECT, REGARDLESS OF THE CONTEXT OF THAT REFERENCE.

THE FOLLOWING EXAMPLES ILLUSTRATE THE OLD AND NEW BEHAVIOUR:

```
      OLD                    NEW

      FN                     FN
VALUE ERROR            VALUE ERROR
      FN                     FN
      ^                      ^

      FN 1                   FN 1
SYNTAX ERROR           VALUE ERROR
      FN 1                   FN 1
      ^                      ^
```

'RESULT ERROR' (EVENT 8) IS NOW REPORTED WHEN AN EXPRESSION WHICH DOES NOT RETURN A RESULT IS USED IN A CONTEXT IN WHICH ONE WAS EXPECTED. FOR EXAMPLE:

```
      OLD                    NEW

      1+⍬'→0'                1+⍬'→0'
VALUE ERROR            RESULT ERROR
⍬     →0               ⍬     →0
      ^                      ^

      ∇ FN A                 ∇ FN A
      ∇                      ∇

      (FN 8)÷2               (FN 8)÷2
VALUE ERROR            RESULT ERROR
      (FN 8)÷2               (FN 8)÷2
       ^                      ^
```

'SYNTAX ERROR' CONTINUES TO BE REPORTED WHEN A STATEMENT DOES NOT CONFORM TO VALID APL SYNTAX RULES.

ITEM NO. 12 JAN 21, 1982

## RELATIONAL FUNCTIONS WORK BETTER

THE BEHAVIOUR OF THE RELATIONAL FUNCTIONS, AND αεω, αιω, α≡ω, FOR FLOATING POINT ARGUMENTS HAS BEEN CHANGED TO REMOVE SOME LONGSTANDING (>15 YEARS) BLEMISHES.

1. IT IS NO LONGER POSSIBLE TO OBTAIN A DOMAIN ERROR FOR ANY PAIR OF NUMERICAL ARGUMENTS. IT USED TO BE, FOR EXAMPLE, THAT (⌈/ι0)=⌊/ι0 CAUSED A DOMAIN ERROR.
2. VERY SMALL ARGUMENTS NO LONGER GIVE INCORRECT RESULTS. IT USED TO BE, FOR EXAMPLE, THAT THE COMPARISON 6E¯79=6.1E¯79 GAVE A TRUE RESULT.
3. ON THE AVERAGE, THE RELATIONAL FUNCTIONS FOR FLOATING POINT AND COMPLEX ARGUMENTS (WITH NONZERO ⎕CT) WILL EXECUTE MORE EFFICIENTLY, SINCE THE COMPLICATED ALGORITHM DESCRIBED IN SATN23 WILL ONLY BE USED IF THE ARGUMENTS ARE UNEQUAL AND OF THE SAME SIGN.

---

ITEM NO. 11 JAN 13, 1982

## NEW ADDRESS FOR I.P. SHARP TORONTO

SINCE I.P. SHARP HAS MOVED TO A NEW OFFICE BUILDING, THE SHIPPING ADDRESS HAS CHANGED TO:

I.P. SHARP ASSOCIATES
19TH FLOOR
EXCHANGE TOWER
2 FIRST CANADIAN PLACE
TORONTO, ONTARIO
M5X 1E3

TELEPHONE (UNCHANGED): (416) 364 5361

---

ITEM NO. 9 DEC 23, 1981

## NEW SYSTEM COMMAND: )XLOAD

A NEW SYSTEM COMMAND, )XLOAD, HAS BEEN ADDED TO THE SHARP APL SYSTEM. )XLOAD LOADS A WORKSPACE <u>WITHOUT</u> CAUSING THE EXECUTION OF ITS LATENT EXPRESSION (⎕LX), IF YOU ARE THE OWNER OF THE WORKSPACE. IF THE USER EXECUTING THE )XLOAD OPERATION IS NOT THE OWNER, THEN THE ERROR MESSAGE 'IMPROPER LIBRARY REFERENCE' IS PRINTED, AND THE ACTIVE WORKSPACE IS NOT AFFECTED. THE OWNER OF A WORKSPACE CAN BE DETERMINED BY EXAMINING (2 ⎕WS 4)[1+⎕IO] .

THE USE OF )XLOAD CAN OBVIATE THE NEED FOR COMPLICATED LATENT EXPRESSIONS WHICH ARE DESIGNED TO HAVE NO EFFECT WHEN THE WORKSPACE IS LOADED BY A PARTICULAR ACCOUNT NUMBER. MOREOVER, WITH )XLOAD, THERE IS OFTEN NO NEED TO SAVE A 'TEMPORARY' ⎕LX IN THE MAINTENANCE VERSION OF A WORKSPACE, SO THAT THE WORKSPACE CAN BE LOADED IN THE SAME STATE IN WHICH IT WAS SAVED.

---

ITEM NO. 8 DEC 17, 1981

ENHANCEMENTS TO □OUT

TWO NEW CATEGORIES OF □OUT OUTPUT HAVE BEEN ADDED.
A) WITH □OUT IN EFFECT, WHEN AN ERROR DISPLAY IS PRINTED AT THE TERMINAL, IT IS ALSO APPENDED TO THE □OUT FILE, PREFACED BY THE CONTROL MESSAGE 0019 0001 QUAD ER

B) INPUT IN IMMEDIATE EXECUTION MODE, QUAD MODE, FUNCTION DEFINITION MODE, AND QUAD-PRIME INPUT MODE
IS APPENDED TO FILE, PREFACED BY ONE OF
0020 0001 INPUT
0021 0001 QUAD
0022 0001 QUAD PRIME
0023 0001 FN DEFN

IF THE INPUT LINE IS REJECTED FOR ONE OF THE FOLLOWING REASONS
THE RELEVANT CONTROL MESSAGE WILL BE APPENDED
0024 0000 WS FULL
0025 0000 ST FULL
0026 0000 DEFN ERR
0027 0000 NOT IN DEFN (I.E. SYSTEM COMMAND ATTEMPTED IN DEFN MODE)
0028 0000 OPEN QUOTE

ITEM NO. 4 NOV 26, 1981
NEW COMMS FACILITIES INTRODUCED NOV 81

1. COMPATIBILITY ALLOWS USERS TO CONTINUE ACCESS EXACTLY AS PREVIOUSLY WITH ONE VERY SMALL EXCEPTION. IF THEY ENTER INVALID HOST SELECTION CHARACTERS AFTER O) THEY WILL RECEIVE 'LINES DOWN' WHERE PREVIOUSLY THEY WOULD HAVE BEEN CONNECTED BY DEFAULT TO IPSA PRODUCTION SYSTEM.

2. IT IS POSSIBLE TO CHOOSE TO HAVE CHARACTERS TRANSMITTED FROM YOUR KEYBOARD RETURNED (ECHOED) TO YOUR PAGE OR SCREEN BY THE NODE TO WHICH YOU ARE DIRECTLY CONNECTED, AS IT FORWARDS THEM TO THE HOST. THIS HAS THE ADVANTAGE OF SHOWING WHETHER OR NOT CHARACTERS WERE CORRUPTED BY THE LOCAL DIAL UP LINE. IN ORDER TO USE THIS FACILITY SET YOUR TERMINAL AND MODEM TO 'FULL DUPLEX' AND START YOUR SIGN ON WITH OE) CARRIAGE RETURN. IN ORDER TO TERMINATE THIS FACILITY, SIGN OFF IN THE NORMAL WAY AND THEN (WITHIN 20 SECONDS OF SIGN OFF) SIGN ON WITH OL) CARRIAGE RETURN. (DONT FORGET TO CHANGE THE TERMINAL BACK TO 'HALF DUPLEX'.) NOTE THAT O) STILL OPERATES AS BEFORE. CHARACTERS ARE ONLY ECHOED DURING INPUT TO APL.

3. IT IS POSSIBLE TO SIGN ON TO ANOTHER HOST IMMEDIATELY AFTER SIGNING OFF. THIS APPLIES TO BOTH HARDWIRE AND DIAL UP CONNECTIONS. THE DIAL UP CONNECTION WILL REMAIN FOR 20 SECONDS AFTER A SIGN OFF TO ALLOW ANOTHER SIGN ON.

4. IF APL IS NOT IN INPUT, THE ASCII CHARACTER ETX (CONTROL C) WILL BE TREATED AS A BREAK. USE OF THE BREAK OR ATTN KEY REMAINS UNCHANGED. ETX IS TREATED AS DATA DURING INPUT TO APL.

5. THERE IS AN IMPROVED CHANCE OF SUCCESSFUL CONNECTION TO THE IPSA PRODUCTION SYSTEM IN THE EVENT OF A FAILED 3705.

6. IF YOU TRANSMIT A BREAK SIGNAL DURING INPUT TO APL, THE LINE WILL BE DISCARDED AND YOU WILL RECEIVE ANOTHER PROMPT FOR INPUT.

NOTES

1. USE OF THE OE) FACILITY WITH DISCS OR CASSETTES IS NOT RECOMMENDED. IF CHARACTERS ARRIVE TOO FAST FOR THE NODE TO RETURN THEM A CHARACTER MAY BE MISSED OUT. SUCH A MISSED CHARACTER WOULD NOT BE SENT TO THE HOST EITHER.

2. REMEMBER, IF WHEN SELECTING A HOST, A RESPONSE OTHER THAN 'LINES DOWN' IS RECEIVED, ENTERING )OFF AND RETURN WILL ENABLE YOU TO IMMEDIATELY SELECT ANOTHER HOST. IF 'LINES DOWN' IS RECEIVED ANOTHER HOST CAN BE SELECTED IMMEDIATELY.

ITEM NO. 3 AUG 20, 1981

)TERM NOIDLES MEANS NO IDLES AT ALL.

)TERM NOIDLES PREVIOUSLY CAUSED A SMALL NUMBER OF IDLES TO BE TRANSMITTED TO THE APL TERMINAL. THIS HAS BEEN FIXED, SO THAT NO IDLES SHOULD BE OBSERVED.

ITEM NO. 168 JUN 15, 1981
PRELIMINARY SUPPORT FOR DIRECT DEFINITION

CHANGES HAVE BEEN MADE TO THE TREATMENT OF α, ω AND ∇ TO FACILITATE DEVELOPMENT OF THE DIRECT DEFINITION OPERATOR. DIRECT DEFINITION WILL BE DISCUSSED IN DETAIL IN A PAPER BY IVERSON AND WOOSTER AT APL 81.

ALPHA AND OMEGA

α AND ω MAY BE USED AS NAMES FOR ANY OBJECT IN A WORKSPACE. THEY DO NOT REQUIRE SURROUNDING BLANKS AND MAY NOT FORM LONGER NAMES. FOR EXAMPLE:

αABCω ⟷ α ABC ω

BLANKS ARE REQUIRED AROUND THESE NAMES IN NAMELISTS:

(⎕PACK 'α ABC ω') NOT (⎕PACK 'αABCω')

DEL

∇ WILL EVENTUALLY SERVE TWO PURPOSES: ITS OLD USE AS THE ESCAPE INTO AND OUT OF FUNCTION DEFINITION, AND A NEW USE AS A DYADIC OPERATOR. THE PRESENT CHANGE MERELY INTRODUCES ITS RECOGNITION AS AN OPERATOR SYMBOL, SO THAT (EXCEPT FOR ITS USE IN ENTERING AND LEAVING FUNCTION DEFINITION) IT WILL INVOKE SYNTAX ERROR RATHER THAN DEFN ERROR. THIS ALLOWS THE ERROR TO BE TRAPPED FOR USE IN MODELLING THE PROPOSED OPERATOR.

ITEM NO. 167 JUN 15, 1981
)CPULIM

SHARP APL NOW SUPPORTS A NEW SYSTEM COMMAND, )CPULIM, WHICH ALLOWS A USER TO SPECIFY BOTH A CPU COMPUTATION LIMIT AND A CPU SESSION LIMIT FOR A T-TASK. ITS SYNTAX IS:

)CPULIM N OR )CPULIM N M

WHERE <N> IS THE COMPUTATION LIMIT AND <M> IS THE SESSION LIMIT FOR THE TASK. THE COMPUTATION LIMIT AFFECTS CPU CONSUMPTION FROM ONE TYPED INPUT TO THE NEXT, AND IS THE SAME LIMIT PREVIOUSLY SUPPORTED BY APL FOR T-TASKS. THE SESSION LIMIT AFFECTS CPU CONSUMPTION BEGINNING WHEN THE LIMIT IS SET. IF ONLY A SINGLE NUMBER IS SPECIFIED IN THE ARGUMENT TO THE COMMAND, THAT NUMBER BECOMES THE NEW COMPUTATION LIMIT AND THE TASK'S SESSION LIMIT IS NOT ALTERED.

WHEN EITHER LIMIT EXPIRES, A TRAPPABLE INTERRUPT (EVENT 1003) IS TRIGGERED, AND THE LIMIT IS REFRESHED (NOT CLEARED). A LIMIT THAT IS NOT TRAPPED INTERRUPTS EXECUTION, BUT DOES NOT CAUSE THE TASK TO TERMINATE. AS WITH ⎕RUN, A LIMIT OF ZERO IS TAKEN TO MEAN NO LIMIT. AT SIGNON, THE DEFAULT COMPUTATION LIMIT IS THE VALUE SPECIFIED BY THE SYSTEM ACCOUNT MANAGER. THE DEFAULT SESSION LIMIT IS ALWAYS ZERO.

ITEM NO. 166 JUN 15, 1981

ITEM NO. 166 JUN 15, 1981
S-TASKS AND A GENERAL INTERFACE TO APL

WHEN YOU SIT AT AN APL TERMINAL AND TYPE INSTRUCTIONS, WHAT YOU TYPE IS RELAYED TO A TERMINAL CONTROLLER, WHICH COLLECTS THE MESSAGE FORMED BY YOUR KEYSTROKES, AND THEN PASSES IT TO THE APL SYSTEM FOR PROCESSING. THE RESPONSES FROM APL GO BACK TO THE TERMINAL CONTROLLER, ARE SUITABLY DIVIDED INTO TRANSMISSIONS, AND THEN FORWARDED TO YOUR TERMINAL. THIS INTERMEDIARY CONTROLLER IS AS DISCREET AS POSSIBLE, AND USUALLY YOU'RE NOT AT ALL AWARE OF IT. YOU THINK OF YOURSELF AS COMMUNICATING DIRECTLY WITH APL.

DURING 1980-81, THE APL DEVELOPMENT GROUP AT IPSA WANTED TO BE ABLE TO COMMUNICATE WITH APL NOT JUST THROUGH THE EXISTING TERMINAL INTERFACE, BUT IN OTHER WAYS AS WELL. A MORE GENERAL INTERFACE WOULD MAKE IT EASIER TO SUPPORT TERMINALS WITH QUITE DIFFERENT STYLES OF INTERACTION, OR TO CONTROL TASKS THAT MAKE NO USE OF A TERMINAL AT ALL. THE RESULT OF THEIR EFFORTS IS INCLUDED IN THE JUNE '81 RELEASE AS AP-1. THE LETTERS STAND FOR ''AUXILIARY PROCESSOR''. IF YOU SHARE A VARIABLE WITH AP-1 (BY AN INSTRUCTION SUCH AS 1 ⎕SVO 'VAR') YOU CAN SEND IT INSTRUCTIONS JUST AS YOU MIGHT TYPE THEM FROM A TERMINAL. SETTING A VALUE OF THE VARIABLE YOU'VE SHARED TRANSMITS A MESSAGE TO AP-1, AND REFERRING TO THE VARIABLE READS AP-1'S REPLY. THE FIRST THING YOU'D NEED TO SEND IT IS A SIGN-ON MESSAGE SO THAT YOU CAN START AN APL TASK. AP-1 PASSES YOUR REQUEST TO THE APL SYSTEM, AND RETURNS TO YOU THE USUAL SYSTEM RESPONSE. IF YOU SUPPLY A VALID SIGN-ON, YOU'LL HAVE A NEW TASK AT YOUR COMMAND. IT'S CALLED AN S-TASK BECAUSE IT COMMUNICATES THROUGH A SHARED VARIABLE RATHER THAN THROUGH THE USUAL TERMINAL CONTROLLER.

AP-1 CAN RECEIVE SHARED VARIABLE OFFERS FROM ALREADY-RUNNING APL TASKS. AN OFFER COMES FROM AN APL WORKSPACE WHICH FOR SOME REASON NEEDS TO START A SEPARATE WORKSPACE, RUNNING CONCURRENTLY, AND UNDER ITS CONTROL. AN S-TASK CAN THUS BE A COMPLETELY INDEPENDENT ENVIRONMENT (LOADING ITS OWN WORKSPACES, RUNNING ITS OWN PROGRAMS) BUT STILL REPORTING TO ITS PARENT TASK.

AN S-TASK IS IN SOME RESPECTS LIKE AN N-TASK; HOWEVER, AN N-TASK CAN NEVER RETURN TO IMMEDIATE EXECUTION, AND MUST DO ALL ITS WORK EXECUTING THE ONE INSTRUCTION THAT STARTED IT. BY CONTRAST, AN S-TASK IS AS FULLY CONTROLLABLE AS A T-TASK. IT CAN BE INTERRUPTED, INTERROGATED, ASKED TO LOAD A NEW WORKSPACE, AND SO ON; IT CAN EVEN BE MADE TO CONTINUE AFTER ITS PARENT TASK SIGNS OFF, AND TO ACCEPT CONTROL AGAIN WHEN THE PARENT RETURNS.

SEE SATN-39 FOR MORE INFORMATION.

ITEM NO. 165 JUN 15, 1981
COMPLEX NUMBERS

WITH THE JUNE '81 RELEASE, SHARP APL IS EXTENDED TO RETURN COMPLEX RESULTS WHERE CALLED FOR, AND TO ACCEPT COMPLEX ARGUMENTS FOR MOST (NOT YET ALL) FUNCTIONS. FOR EXAMPLE, MULTIPLICATION AND ADDITION ARE DEFINED ON COMPLEX NUMBERS. TRANSLATIONS AND TRANSFORMATIONS COMMONLY REQUIRED IN GRAPHICS SOFTWARE CAN NOW BE WRITTEN DIRECTLY, WITHOUT THE NEED FOR THE TRIGONOMETRY PREVIOUSLY REQUIRED.

SEVERAL FUNCTIONS HAVE BEEN ADDED EXPRESSLY FOR CONVERSION

BETWEEN VARIOUS REPRESENTATIONS OF A COMPLEX NUMBER. THEY ARE OBTAINED BY NEW LEFT ARGUMENTS TO THE ○ FAMILY OF FUNCTIONS. THE EXPRESSION 9 11○.○ω REPRESENTS ω BY ITS REAL AND IMAGINARY PARTS, WHILE 10 12○.○ω REPRESENTS ω IN POLAR COORDINATES, BY ITS MODULUS AND PHASE. FOR THE INVERSE TRANSFORMATION, AN ARRAY WHOSE LEADING AXIS HAS LENGTH TWO AND CONSISTS OF THE REAL AND IMAGINARY PARTS IS RESOLVED INTO COMPLEX NUMBERS BY ¯10 ¯11+.○ω, AND A REPRESENTATION CONSISTING OF MODULUS AND PHASES BY ¯9 ¯12×.○ω.

COMPARISONS OF ORDER ARE UNDEFINED ON COMPLEX NUMBERS, SO THE RELATIONS < ≤ > ≥ AND GRADING FUNCTIONS ⍋ AND ⍒ DO NOT INCLUDE COMPLEX NUMBERS IN THEIR DOMAIN. FLOOR AND CEILING ARE DEFINED, BUT HAVE NOT BEEN INCLUDED IN THE RELEASE BECAUSE OF REMAINING TECHNICAL QUESTIONS ABOUT THEIR DEFINITIONS.

SEE SATN-40 FOR MORE INFORMATION.

ITEM NO. 164 JUN 15, 1981
OPERATORS

FROM ITS EARLIEST DAYS, APL HAS INCLUDED BOTH FUNCTIONS AND OPERATORS. BUT THE DISTINCTION BETWEEN THEM WASN'T RECOGNIZED AT FIRST, AND THE WORDS WERE USED INTERCHANGEABLY. STARTING ABOUT TEN YEARS AGO, THEORETICIANS REALIZED THAT ''FUNCTION'' SHOULD BE RESERVED FOR TRANSFORMATIONS OF DATA. THE WORD ''OPERATOR'' WAS RESERVED FOR TRANSFORMATIONS THAT PRODUCE NEW FUNCTIONS (WHICH CAN THEN BE EXECUTED). IN THE EXPRESSION +/[1]A THE SYMBOL / DENOTES AN OPERATOR THAT TAKES THE FUNCTION + AND TURNS IT INTO SUMMATION, AND [1] IS AN OPERATOR THAT TAKES THE NUMBER 1 AND THE FUNCTION SUMMATION AND PRODUCES FIRST-AXIS SUMMATION. ONLY AFTER + HAS UNDERGONE THOSE TWO TRANSFORMATIONS IS IT EXECUTED ON THE DATA A.

GRADUALLY IT BECAME CLEAR THAT THE ABILITY TO USE OPERATORS TO CREATE FUNCTIONS WAS ONE OF APL'S MOST POWERFUL ATTRIBUTES. THAT WAS THE SUBJECT OF KEN IVERSON'S PAPER HE PRESENTED WITH BOB BERNECKY AT THE 1980 APL USERS MEETING IN TORONTO.

THE JUNE '81 RELEASE CONTAINS THREE NEW OPERATORS, THE FIRST ADDED TO APL SINCE THE SCAN OPERATOR WAS INTRODUCED IN 1973. IT HAS TAKEN A LONG TIME FOR A THEORY OF OPERATORS TO EMERGE, BUT NOW THAT THE FOUNDATIONS ARE LAID, OTHER OPERATORS WILL FOLLOW. THIS RELEASE CONTAINS ONLY A FEW OF THOSE ALREADY PROPOSED. THE STRATEGY HAS BEEN TO START SLOWLY, WITH THOSE THAT ARE ALREADY WELL UNDERSTOOD, AND TO ADD OTHERS LATER AS EXPERIENCE WITH THEM ACCUMULATES. THE THREE CHOSEN ARE PARTICULARLY USEFUL IN EXPLOITING ENCLOSED ARRAYS.

THE THREE NEW OPERATORS ARE VARIOUS FORMS OF COMPOSITION. THAT IS, THEY CONCERN HOW TWO FUNCTIONS ARE USED TOGETHER IN PROCESSING THE ELEMENTS OF AN ARRAY (OR A PAIR OF ARRAYS). FOR EXAMPLE, THE EXPRESSION ⍴⍤⊃> Y CALCULATES THE SHAPE THAT EACH ELEMENT OF Y WOULD HAVE WHEN DISCLOSED. THAT'S NOT THE SAME AS ⍴>Y. THE EXPRESSION ⍴>Y WOULD MEAN ''FIRST DISCLOSE ALL OF Y; THEN FIND THE SHAPE OF THAT RESULT. BY CONTRAST, ⍴⍤⊃> Y DISCLOSES AN ELEMENT OF Y AND IMMEDIATELY FINDS ITS SHAPE. THAT DOUBLE SEQUENCE IS REPEATED FOR ALL OF THE ELEMENTS OF Y.

THE COMPOSED FUNCTIONS PRODUCED BY ANY OF THE COMPOSITION OPERATORS IS APPLIED TO EACH CELL OF THE ARGUMENT. A CELL IS THE MINIMUM SHAPE ON WHICH THE FUNCTION ω (THAT IS, THE FUNCTION TO THE RIGHT OF THE OPERATOR SYMBOL) IS DEFINED. WHEN ω IS DEFINED

ON SCALARS, THE CELL IS A SCALAR, AND SO THE COMPOSITION APPLIES TO EACH SCALAR. WHEN ω IS DEFINED ON A VECTOR (FOR EXAMPLE, ⊥) A CELL IS A VECTOR, AND SO THE COMPOSITION APPLIES TO EACH VECTOR ON THE LAST AXIS OF Y. THE AXES OF Y THAT ARE NOT USED BY THE CELLS ARE CALLED THE FRAME. THE COMPOSITION IS EXECUTED INDEPENDENTLY FOR EACH CELL OF THE FRAME. THE CONCEPTS OF ''FRAME'' AND ''CELL'', TAKEN TOGETHER WITH COMPOSITION, PERMIT A SIMPLE STATEMENT OF MANY PROCEDURES THAT OTHERWISE WOULD HAVE TO BE WRITTEN WITH EXPLICIT ITERATION.

AT PRESENT, THE OPERATORS ⍠ ⍤ AND ⍥ ARE IMPLEMENTED ONLY FOR A SUBSET OF FUNCTIONS. THEY DON'T YET APPLY TO USER-DEFINED FUNCTIONS, OR TO DERIVED FUNCTIONS (SUCH AS +/). BUT THEY DO APPLY TO ENCLOSE AND DISCLOSE, WITH MANY USEFUL CONSEQUENCES.

THE OPERATOR ⍥ IS CALLED OVER. FOR EXAMPLE, FOR TWO CONFORMABLE ARRAYS X AND Y, TO FIND THE RECIPROCAL OF THE SUMS OF CORRESPONDING ELEMENTS, YOU COULD WRITE X ÷⍥+ Y. BECAUSE + IS A SCALAR FUNCTION, YOU COULD GET THE SAME RESULT BY ÷X+Y. BUT WHEN ⍥ IS EXTENDED TO NON-SCALAR FUNCTIONS (IT HASN'T BEEN YET) YOU COULD, FOR EXAMPLE, GENERATE INDEPENDENT SETS OF RANDOM DEALS, EACH HAVING A LENGTH UNRELATED TO THE LENGTH OF THE OTHERS, BY THE EXPRESSION X <⍥? Y. FOR EACH PAIR OF ELEMENTS, ONE FROM X AND ONE FROM Y, X_ ? Y_ IS EVALUATED AND IMMEDIATELY ENCLOSED. (HERE X_ IS A NON-APL SHORTHAND FOR ''A CELL OF X''.)

EACH OF THE COMPOSITION OPERATORS GENERATES A DERIVED FUNCTION THAT IS AMBIVALENT. THAT IS, THE DERIVED FUNCTION IS UNDERSTOOD TO BE EITHER DYADIC OR MONADIC, DEPENDING ON THE CONTEXT. THE EXAMPLE X α⍥ω Y SHOWED A DYADIC USE, MEANING ''FOR EACH PAIRING OF X AND Y, EXECUTE α X_ ω Y_''. BECAUSE THERE ARE TWO ARGUMENTS, X AND Y, THE DYADIC MEANING OF FUNCTION ω IS UNDERSTOOD. BUT IF THE EXPRESSION WERE α⍥ω Y, WITH NO LEFT ARGUMENT, THE MONADIC MEANING OF ω WOULD BE UNDERSTOOD.

THE OPERATOR ⍤ IS CALLED ON. IT DIFFERS FROM ⍥ (OVER) ONLY IN ITS DYADIC USE. WITH α⍥ω, THE AMBIVALENT FUNCTION IS THE ONE ON THE RIGHT, WHEREAS WITH α⍤ω, THE LEFT FUNCTION α IS AMBIVALENT, AND ω IS MONADIC. IN AN EXPRESSION SUCH AS X α⍤ω Y, EACH CELL OF THE RESULT IF FORMED BY APPLYING THE MONADIC FUNCTION ω SEPARATELY TO A CELL OF X AND A CELL OF Y, AND THEN EXECUTING THE DYADIC FUNCTION α. FOR EXAMPLE, TO FIND THE SUM OF THE LOGS OF X AND Y, YOU COULD WRITE X +⍤⍟ Y. TO FORM AN ARRAY BY DISCLOSING CORRESPONDING ELEMENTS OF X AND Y AND CATENATING EACH OF THE PAIRS OF ARRAYS THUS FORMED: X ,⍤> Y.

THE OPERATOR ⍠ IS CALLED WITH. LIKE ⍤, WITH APPLIES THE MONADIC FUNCTION ω TO EACH CELL OF ITS ARGUMENTS, AND THEN THE FUNCTION α TO TO EACH OF THOSE RESULTS. BUT ⍠ GOES FURTHER. FOR EACH ARGUMENT CELL, AFTER EXECUTING THE MONADIC FUNCTION ω, IT RE-TRANSFORMS THAT RESULT BY THE INVERSE OF ω. THIS IS A SEQUENCE THAT TURNS UP IN A GREAT MANY SITUATIONS. YOU CONVERT THE ARGUMENTS BEFORE APPLYING SOME FUNCTION, AND THEN CONVERT BACK THE RESULTS. IF YOU WERE COMPUTING THE SUMS OF THE LOGS, FOR INSTANCE, THE NEXT STEP IS PROBABLY TO CONVERT BACK BY FINDING THE ANTILOG OF THE RESULT. X +⍠⍟ Y DOES EXACTLY THAT. IT IS EQUIVALENT TO *(⍟X)+(⍟Y).

A MAJOR USE FOR ⍠ ARISES WITH ENCLOSED ARRAYS. YOU WANT TO DISCLOSE EACH ELEMENT OF THE ARGUMENT, DO SOMETHING TO IT (OR THEM) AND THEN RE-ENCLOSE THE RESULT. IN MANY SITUATIONS IT'S ESSENTIAL TO RE-ENCLOSE EACH CELL OF THE RESULT BECAUSE THEY VARY IN SHAPE, AND HENCE CAN BE FITTED INTO AN ARRAY RESULT ONLY WHEN ENCLOSED. FOR EXAMPLE, SUPPOSE X IS AN ARRAY IN WHICH EACH ELEMENT IS SOMEBODY'S FIRST NAME (AS AN ENCLOSED VECTOR), AND Y IS A SIMILAR ARRAY OF SURNAMES. EACH NAME HAS ITS OWN LENGTH. YOU

JOIN EACH FIRST NAME TO ITS CORRESPONDING SURNAME BY X ,"> Y. TO EXECUTE WITH, THE FUNCTION ⍵ MUST HAVE A RECOGNIZED INVERSE; AT PRESENT, THE SYSTEM RECOGNIZES INVERSES FOR ENCLOSE, DISCLOSE, POWER AND LOG, AND FOR NOT, RECIPROCAL, TRANSPOSE, AND NEGATIVE (WHICH ARE SELF-INVERSE).

SEE SATN-41 FOR MORE INFORMATION.

ITEM NO. 163 JUN 15, 1981
ENCLOSED ARRAYS

IN APL, A DATA OBJECT IS A RECTANGULAR ARRAY (UNLESS IT'S A PACKAGE, WHICH IS STRUCTURED MORE LIKE A WORKSPACE). AN ARRAY CAN HAVE ANY NUMBER OF AXES; EACH AXIS CAN HAVE ANY LENGTH. THE SIMPLEST ARRAY IS A SCALAR, WITH NO AXES AT ALL, AND HENCE NO STRUCTURE. EACH ELEMENT OF AN ARRAY IS A SCALAR. UP TO NOW, ARRAYS HAVE BEEN OF TWO TYPES: NUMERIC AND CHARACTER. IN A NUMERIC ARRAY, EACH ELEMENT IS A SINGLE NUMBER. IN A CHARACTER ARRAY, EACH ELEMENT IS A SINGLE CHARACTER.

THE NEW FUNCTION ENCLOSE CONVERTS ANY ARRAY INTO A NEW KIND OF SCALAR. SUPPOSE YOU HAVE A NUMERIC ARRAY N AND A CHARACTER ARRAY C. YOU COULD ENCLOSE EACH OF THEM AND FORM A TWO-ELEMENT VECTOR X:

```
Z←(<N), <C
```

THE ARRAY Z IS NEITHER NUMERIC NOR CHARACTER, BUT BELONGS TO THE NEW DATA-TYPE: ENCLOSED.

THE FUNCTION > DISCLOSE REVERSES THE PROCESS. FOR EXAMPLE, >Z[1] GIVES YOU THE VALUE OF THE ARRAY N, WHILE >Z[2] GIVES YOU THE VALUE OF C. GENERALLY YOU CAN DISCLOSE ONLY ONE SCALAR AT A TIME. HOWEVER, YOU CAN DISCLOSE SEVERAL AT ONCE IF, WHEN DISCLOSED, THEY ALL HAVE THE SAME TYPE AND SHAPE. FOR EXAMPLE, IF EACH OF THE FIRST THREE ELEMENTS OF AN ARRAY W IS A NUMERIC MATRIX WITH SHAPE 24 30, THEN >W[⍳3] GIVES YOU A NUMERIC ARRAY HAVING SHAPE 3 24 30.

ALL THE STRUCTURAL FUNCTIONS, SUCH AS ⍴ ↑ ↓ ⌽ ⍉ , [] ⍳ ETC., APPLY TO AN ENCLOSED ARRAY JUST AS THEY DO TO ANY OTHER. FUNCTIONS THAT INVOLVE COMPUTATION (SUCH AS + × ⊥ ETC.) CAN'T BE APPLIED DIRECTLY TO AN ENCLOSED ARRAY. YOU HAVE TO CONSTRUCT AN EXPRESSION THAT DISCLOSES EACH OF THE ELEMENTS IN TURN AND DOES THE COMPUTATION ON THE DISCLOSED VALUE. THREE NEW FUNCTION COMPOSITION OPERATORS, ON, WITH AND OVER, ALSO INTRODUCED WITH THIS RELEASE AND DESCRIBED ELSEWHERE, MAKE THAT EASY TO DO.

THE NEW FUNCTION MATCH (WRITTEN A≡B) ISN'T SPECIFIC TO ENCLOSED ARRAYS, BUT SIMPLIFIES WORKING WITH THEM. THE RESULT OF A≡B IS A SCALAR 1 WHEN A IS IDENTICAL TO B IN EVERY RESPECT, AND OTHERWISE IS A SCALAR 0. APPLIED TO ENCLOSED ARRAYS, ''IDENTICAL'' MEANS THAT THE TWO ARRAYS HAVE THE SAME RANK AND SHAPE, AND THAT EACH ELEMENT OF ONE, WHEN DISCLOSED, IS IDENTICAL TO THE CORRESPONDING DISCLOSED VALUE OF THE OTHER. A SIMPLE ARRAY (THAT IS, ONE WHOSE ELEMENTS ARE NOT ENCLOSED) IS NEVER EQUAL TO AN ENCLOSED ARRAY. THAT IS, < PRODUCES AN ENCLOSED SCALAR EVEN WHEN THE THING TO BE ENCLOSED IS ALREADY A SCALAR (SAY A SCALAR NUMBER OR CHARACTER). IN ARRAY THEORY, THAT'S CALLED STRICT ENCLOSE. HOWEVER, DISCLOSING A SCALAR THAT'S ALREADY SIMPLE (THAT IS, CHARACTER OR NUMERIC, BUT NOT ENCLOSED) GIVES YOU THE SCALAR UNCHANGED; THAT'S

CALLED PERMISSIVE DISCLOSE.

THE IMPLICATIONS OF BEING ABLE TO ENCLOSE AN ARRAY HAVE LONG BEEN A TOPIC OF DEBATE IN THE APL COMMUNITY. SEVERAL THEORETICAL ISSUES REMAIN UNRESOLVED. THE STRATEGY BEHIND THIS RELEASE OF ENCLOSED ARRAYS HAS BEEN TO START CAUTIOUSLY, PROVIDING FEATURES FOR WHICH THERE IS SOLID AGREEMENT, AND DEFERRING UNTIL SUBSEQUENT RELEASES OTHERS WHICH, WHILE ATTRACTIVE, HAVE NOT YET ACHIEVED AGREEMENT. THUS ENCLOSED ARRAYS WILL PROBABLY BE EXTENDED IN FUTURE RELEASES. SEE SATN-41 FOR MORE INFORMATION.

ITEM NO. 162 JUN 15, 1981
NEW RELEASE OF SHARP APL

SIGNIFICANT FUNCTIONAL AND PERFORMANCE ENHANCEMENTS ARE BEING INTRODUCED WITH THE INSTALLATION OF THE JUNE 1981 VERSION OF SHARP APL. THESE ARE SUMMARIZED BELOW. MORE DETAILED INFORMATION APPEARS IN ADDITIONAL NEWS ITEMS. NEW SATN'S WILL BE AVAILABLE ONLINE AT THE TIME OF THE INSTALLATION (JUNE 20) AND FROM YOUR LOCAL I.P. SHARP BRANCH OFFICE SOON THEREAFTER.

IN ORDER TO INSTALL THE NEW SOFTWARE, WE WILL BE SHUTTING DOWN OUR TORONTO APL SERVICE AT 0200 UTC ON JUNE 20.

## SUMMARY OF ENHANCEMENTS

### NEW FUNCTIONS AND OPERATORS

THREE NEW PRIMITIVE FUNCTIONS ARE NOW SUPPORTED:

| | |
|---|---|
| ENCLOSE: | Z←<ω |
| DISCLOSE: | Z←>ω |
| MATCH: | Z←α≡ω |

THREE NEW OPERATORS, PROVIDING THREE FORMS OF FUNCTION COMPOSITION, HAVE BEEN INTRODUCED:

| | |
|---|---|
| WITH: | Z←α⍤ω |
| ON: | Z←α⍛ω |
| OVER: | Z←α⍥ω |

THE NEW FUNCTIONS ARE DESCRIBED IN ITEM 163 AND THE NEW OPERATORS ARE DESCRIBED IN ITEM 164. BOTH ARE DEALT WITH IN MORE DETAIL IN SATN-41.

### COMPLEX NUMBERS

COMPLEX NUMBERS ARE NOW SUPPORTED. EXTENSIONS HAVE BEEN MADE TO THE PRIMITIVE FUNCTIONS TO SUPPORT COMPLEX ARGUMENTS AND TO RETURN COMPLEX RESULTS WHERE APPROPRIATE. A BRIEF DESCRIPTION APPEARS IN ITEM 165, AND A DETAILED DESCRIPTION IS AVAILABLE IN SATN-40.

### S-TASKS

THE S-TASK INTERFACE, FIRST DESCRIBED IN THE I.P. SHARP 1980 USERS' MEETING PAPER 'SHARP APL MULTIPROCESSING AND SHARED VARIABLES,' IS NOW AVAILABLE. THIS FACILITY ALLOWS THE USER

TO CONSTRUCT A SHARED VARIABLE INTERFACE TO APL WHOSE VALUES ARE CHARACTER VECTORS WHICH REPRESENT LINES OF INPUT AND OUTPUT. THUS ONE TASK MAY NOW 'SIGNON' ANOTHER TASK, PROVIDE INPUT AND EXAMINE THE RESULTING OUTPUT. ITEM 166 AND SATN-39 CONTAIN MORE DETAILED INFORMATION.

## INNER PRODUCT SPEEDUPS

THE INNER PRODUCT OPERATOR HAS BEEN REWRITTEN TO ACHIEVE PERFORMANCE GAINS. DEPENDING UPON THE FUNCTIONS APPLIED AND THE DATATYPES OF THE ARGUMENTS, AN IMPROVEMENT OF FROM 2-5 TIMES IS OBSERVABLE ON NON-BOOLEAN INNER PRODUCTS. FOR BOOLEAN INNER PRODUCTS, IMPROVEMENTS OF 100 TO 200 TIMES ARE TYPICAL. THE INNER PRODUCTS ∧.= AND ∨.≠ WERE ALREADY HIGHLY OPTIMIZED, AND HAVE NOT CHANGED SUBSTANTIALLY.

## )CPULIM

A NEW SYSTEM COMMAND, )CPULIM, ALLOWS A USER TO SPECIFY BOTH A CPU COMPUTATION LIMIT AND A CPU SESSION LIMIT FOR A T-TASK. EXPIRATION OF EITHER LIMIT MAY BE TRAPPED (EVENT 1003). IF NOT TRAPPED, EXPIRATION WILL INTERRUPT, BUT NOT TERMINATE, THE USER'S SESSION. ITEM 167 CONTAINS MORE DETAILS.

## PRELIMINARY SUPPORT FOR DIRECT DEFINITION

CHANGES HAVE BEEN MADE TO THE TREATMENT OF α, ω, AND ∇ TO FACILITATE FUTURE DEVELOPMENT OF THE DIRECT DEFINITION OPERATOR. DETAILS OF THE PROPOSED NEW OPERATOR WILL BE PRESENTED IN A PAPER BY IVERSON AND WOOSTER AT APL 81. ITEM 168 CONTAINS MORE DETAILS ON THE CHANGES TO α, W, AND ∇.

ITEM NO. 155 NOV 07, 1980
CHANGES TO THE WORKSPACE '1 BTASKREQ' EFFECTIVE NOVEMBER 8TH, 1980

BEGINNING NOVEMBER 8TH, 1980, AN ENHANCED VERSION OF THE BTASK REQUEST WORKSPACE '1 BTASKREQ' WILL BE AVAILABLE. THE FUNCTIONS IN THIS WORKSPACE NOW SUBMIT REQUESTS AND PROVIDE REQUEST STATUS INFORMATION THROUGH A SHARED VARIABLE INTERFACE TO A SPECIAL SYSTEM TASK. THE NEW INTERFACE FUNCTIONS ARE UNLOCKED AND CONSUME SIGNIFICANTLY LESS CPU TIME THAN THEIR PREDECESSORS. USERS ARE ENCOURAGED TO INCORPORATE THESE NEW FUNCTIONS INTO THEIR APPLICATIONS PACKAGES AS SOON AS POSSIBLE.

THE CHANGES ARE:

1. THE FUNCTIONS NOW HAVE NAMES PREFIXED WITH THE LETTERS 'BT' (E.G. <QUEUED> IS NOW NAMED <BTQUEUED>). IN ADDITION, THE <AUTOBREQ> FUNCTION IS NOW NAMED <BTSUBMIT>.

2. THE REQUEST STATUS 'LAPSED' HAS BEEN EXPANDED INTO THREE DISTINCT STATES: 'COMPLETED', 'WITHDRAWN', AND 'FAILED'. THREE NEW FUNCTIONS, <BTCOMPLETED>, <BTFAILED>, AND <BTWITHDRAWN>, ARE PROVIDED. EACH OF THESE FUNCTIONS IS NILADIC AND RETURNS A VECTOR OF REQUEST NUMBERS AS ITS RESULT. THE 'STATUS' FIELD IN THE OUTPUT OF THE <BTDISPLAY> FUNCTION MAY CONTAIN ONE OF THESE NEW CODES. IN ADDITION, THE SECOND COLUMN OF THE RESULT OF THE <INQ> FUNCTION MAY NOW INCLUDE A NUMERIC CODE REPRESENTING ONE OF THESE NEW STATES.

   THE SIGNIFICANCE OF EACH STATE AND THE CORRESPONDING INQ STATUS CODES ARE AS FOLLOWS:

   COMPLETED - THE BTASK WAS SUCCESSFULLY STARTED AND IS NO LONGER IN AN ACTIVE STATE (INQ CODE 3).

   FAILED - THE SCHEDULER COULD NOT SUCCESSFULLY START THE BTASK (INQ CODE 6).

   WITHDRAWN - THE REQUESTOR WITHDREW THE REQUEST (INQ CODE 4).

   AS BEFORE, THE INQ STATUS CODES FOR QUEUED, ACTIVE, AND SYSTEM ERROR ARE 1, 2, AND 5 RESPECTIVELY.

   THE FUNCTION <BTLAPSED> RETURNS THE NUMBERS OF ALL REQUESTS WHOSE STATUS IS ANY ONE OF 'COMPLETED', 'FAILED', OR 'WITHDRAWN'. THIS RESULT IS CONSISTENT WITH THE RESULT OF THE OLD <LAPSED> FUNCTION. NOTE, HOWEVER, THAT AN <INQ> STATUS CODE OF 3 FORMERLY INDICATED THE 'LAPSED' STATUS, BUT NOW IS USED TO DENOTE THE 'COMPLETED' STATUS. EXISTING PROGRAMS WHICH USE THE OLD <INQ> FUNCTION AND MAKE THE ASSUMPTION THAT ALL LAPSED TASKS HAVE A STATUS CODE OF 3 MUST BE MODIFIED IF THEY ARE TO USE <BTINQ>.

3. THE OUTPUT OF THE <BTDISPLAY> FUNCTION CONTAINS ADDITIONAL

INFORMATION ABOUT A REQUEST IF ITS STATUS IS 'ACTIVE' OR 'FAILED'. IF THE STATUS OF THE REQUEST IS 'ACTIVE', THE TASKID OF THE ACTIVE BTASK WILL BE GIVEN. IF THE STATUS OF THE REQUEST IS 'FAILED', THE WORD 'FAILED' IS FOLLOWED BY AN ERROR MESSAGE WHICH EXPLAINS WHY THE BTASK COULD NOT BE STARTED (E.G. LOADWSID NOT FOUND).

ITEM NO. 154 NOV 04, 1980

A NEW RELEASE OF SHARP APL WILL BE INSTALLED NOVEMBER 8TH, 1980

THE CHANGES ARE AS FOLLOWS:

1. THE FUNCTIONS ∧ AND ∨ HAVE BEEN EXTENDED TO BE THE LEAST COMMON MULTIPLE (LCM) AND THE GREATEST COMMON DIVISOR (GCD) FUNCTIONS. THIS EXTENSION IS DISCUSSED IN DETAIL IN NEWS ITEM 153.

2. THE RESULT OF ⍞ INPUT IS ALWAYS A VECTOR. PREVIOUSLY, THE RESULT WAS A SCALAR IF A SINGLE CHARACTER WAS ENTERED.

3. ⎕PW NO LONGER LIMITS THE LENGTH OF A LINE WHICH CAN BE EDITED. A LINE LONGER THAN 250 CHARACTERS CANNOT BE EDITED.

4. AN INPUT LINE RECALLED FOR EDITING VIA THE ) COMMAND IS PREFIXED BY SIX SPACES IF IN IMMEDIATE EXECUTION, OR BY THE CURRENT LINE NUMBER IF IN FUNCTION DEFINITION.

5. THE LENGTH OF AN INPUT LINE IS LIMITED BY THE AVAILABLE SPACE IN THE WORKSPACE, RATHER THAN BY THE PREVIOUS FIXED LIMIT OF 512 CHARACTERS. HENCE, IT IS POSSIBLE TO RECEIVE A NON-TRAPPABLE 'WS FULL' IN RESPONSE TO AN INPUT. HOWEVER, THE SYSTEM GUARANTEES SUFFICIENT SPACE TO DELETE FUNCTION LINES WHEN IN FUNCTION DEFINITION, OR TO ENTER A SYSTEM COMMAND OF LESS THAN 150 CHARACTERS. WHEN ENTERING A VERY LONG INPUT IT IS POSSIBLE FOR THE DYNAMIC, INTERMEDIATE TYPEWRITER BUFFERS TO BE EXHAUSTED. THIS RESULTS IN A 'RESEND' MESSAGE.

6. UNMATCHED QUOTES RESULT IN THE MESSAGE 'EDIT OPEN QUOTE', FOLLOWED BY A DISPLAY OF THE LINE, AND ENTRY INTO LINE-EDITING MODE WITH THE CURSOR POSITIONED IN THE MIDDLE OF THE LINE.

7. FUNCTION LINES CAN ONLY BE DELETED WITH THE [Δ... ] SEQUENCE. LINEFEED-CARRIAGE RETURN WILL NO LONGER DELETE A FUNCTION LINE.

8. WHEN THE RIGHTMOST CHARACTERS OF AN INPUT LINE ARE BLANKS (WHETHER ENTERED EXPLICITLY OR FROM A PROMPT), BACKSPACING ERASES THEM.

9. ANY LINE-EDITING SEQUENCE CAN BE CANCELLED WITH O BACKSPACE U BACKSPACE T.

10. THERE ARE TWO NEW, NON-TRAPPABLE ERROR MESSAGES.

ENTRY ERROR - THE INPUT LINE CONTAINED A NON-APL CHARACTER (USUALLY AN ILLEGAL OVERSTRIKE). 'CHAR ERROR' WILL INDICATE A VALID APL CHARACTER IN AN INVALID CONTEXT, SUCH AS A ⎕TRAP EXPRESSION CONTAINING A BACKSPACE.

LINE TOO LONG - INDICATES AN ATTEMPT TO EDIT A LINE LONGER THAN 250 CHARACTERS, OR TO CREATE AN EDITED LINE LONGER THAN 250 CHARACTERS.

ALL BUT THE FIRST OF THESE CHANGES RESULT FROM A REDESIGN OF PARTS OF THE SYSTEM IN PREPARATION FOR SUPPORTING S-TASKS IN A SUBSEQUENT RELEASE IN THE MIDDLE OF 1981. AN S-TASK IS A NEW TYPE OF APL TASK THAT USES A SHARED VARIABLE, INSTEAD OF A TERMINAL, FOR COMMUNICATING WITH A USER. S-TASKS ARE DESCRIBED IN RICHARD LATHWELL'S PAPER, 'SHARP APL MULTIPROCESSING AND SHARED VARIABLES', IN THE PROCEEDINGS OF THE 1980 APL USERS MEETING.

ITEM NO. 153 NOV 04, 1980
LCM/GCD FUNCTIONS EXTEND AND/OR

SEVERAL YEARS AGO IT WAS PROPOSED [1] THAT THE LOGICAL FUNCTIONS 'AND' (∧) AND 'OR' (∨) BE EXTENDED TO APPLY TO ARBITRARY NUMERIC ARGUMENTS. THIS CHANGE WILL BE EFFECTIVE NOVEMBER 8TH, 1980. WITH THIS EXTENSION THE SYMBOLS ∧ AND ∨ DENOTE THE LEAST COMMON MULTIPLE (LCM) AND GREATEST COMMON DIVISOR (GCD) FUNCTIONS, RESPECTIVELY. THEIR DEFINITIONS, WHEN APPLIED TO BOOLEAN ARGUMENTS, REMAIN UNCHANGED.

THE GCD OF TWO NUMBERS IS THE LARGEST NUMBER WHICH DIVIDES BOTH OF THEM EVENLY. FOR EXAMPLE, 13 IS THE LARGEST NUMBER WHICH DIVIDES BOTH 65 AND 78, AND IS THUS THEIR GCD. THE GCD OF TWO NON-NEGATIVE NUMBERS IS LESS THAN OR EQUAL TO THE SMALLER OF THE TWO NUMBERS.

THE LCM OF TWO NUMBERS IS THEIR PRODUCT DIVIDED BY THEIR GCD. FORMALLY,

$$A \wedge B \leftrightarrow A \times B \div A \vee B$$

THIS MEANS THAT THE LCM OF TWO POSITIVE NUMBERS IS THE SMALLEST POSITIVE NUMBER THAT THEY BOTH DIVIDE EVENLY. FOR EXAMPLE, 126 IS THE SMALLEST POSITIVE NUMBER DIVISIBLE BY BOTH 42 AND 63, AND IS THUS THEIR LCM. THE LCM OF TWO POSITIVE NUMBERS IS LARGER THAN OR EQUAL TO THE LARGER OF THE TWO NUMBERS.

THE RESULT OF GCD IS ZERO IF BOTH ARGUMENTS ARE ZERO, OTHERWISE IT IS POSITIVE.

THE RESULT OF LCM IS ZERO IF EITHER ARGUMENT IS ZERO, OTHERWISE IT IS POSITIVE IF THE SIGNS OF THE ARGUMENTS AGREE.

GENERAL DISCUSSION

ALTHOUGH THE NOTIONS OF LCM AND GCD ARE RELATIVELY EASY TO DESCRIBE, THERE ARE SEVERAL DETAILS WHICH REQUIRE SOME MORE EXPLANATION.

1. THE LCM AND GCD FUNCTIONS ARE SCALAR DYADIC FUNCTIONS, AND CAN BE USED IN REDUCTION, SCAN, AND INNER AND OUTER PRODUCT.

2. THE LCM AND GCD FUNCTIONS ARE DEFINED FOR ARBITRARY NUMERIC VALUES.

3. THE IDENTITY ELEMENT FOR LCM IS 1 AND FOR GCD IS 0, JUST AS FOR THE 'AND' AND 'OR' FUNCTIONS. STRICTLY SPEAKING, HOWEVER, 1 IS AN IDENTITY ELEMENT FOR LCM ONLY FOR INTEGRAL ARGUMENTS. FOR EXAMPLE, 1∧0.8 IS 4. FURTHERMORE, 0 IS AN IDENTITY ELEMENT FOR GCD ONLY WHEN THE OTHER ARGUMENT IS NON-NEGATIVE. FOR EXAMPLE, 0∨¯8 IS 8.

4. BOTH LCM AND GCD ARE DEPENDENT ON ⎕CT. THIS MEANS THAT IT IS POSSIBLE TO GET ⎕CT ERROR WHEN USING ∧ AND ∨.

[1] MCDONNELL, E.E., 'A NOTATION FOR THE GCD AND LCM FUNCTIONS', PROCEEDINGS OF THE APL 75 CONGRESS.

ITEM NO. 152 SEP 19, 1980
UPGRADE AND DOWNGRADE EXTENDED

MONADIC GRADE HAS BEEN EXTENDED TO ALL NON-SCALAR NUMERIC ARRAYS.

DYADIC GRADE HAS BEEN INTRODUCED TO ALLOW SORTING OF CHARACTER DATA. DYADIC GRADE IS BASED UPON THE DEFINITION PROPOSED BY HOWARD J. SMITH AT APL79.

FOR DETAILS SEE SATN 35.

ITEM NO. 151 SEP 11, 1980
1980 APL USERS MEETING, OCTOBER 6-8, TORONTO

A MAIL STRIKE IN CANADA IS AFFECTING RECEIPT OF REGISTRATIONS FOR THE APL USERS MEETING.

IF YOU HAVE NOT YET REGISTERED AND WOULD LIKE TO DO SO, YOU COULD GIVE REGISTRATION DETAILS TO YOUR LOCAL SHARP OFFICE, OR CONTACT ROSANNE WILD BY PHONE AT (416) 364-5361 OR VIA MAILBOX - (RWI).

IF YOU HAVE ALREADY SENT IN YOUR REGISTRATION FORM AND HAVE NOT RECEIVED A CONFIRMATION BACK FROM I.P. SHARP, IT IS AGAIN ADVISED THAT YOU CONTACT ROSANNE WILD OR YOUR LOCAL SHARP OFFICE.

THE 1980 APL USERS MEETING PROMISES TO BE THE LARGEST INTERNATIONAL

APL GATHERING OF THE YEAR - REGISTER SOON!

ITEM NO. 150 AUG 31, 1980
ABSTRACTS FOR APL 81 DUE OCTOBER 1ST

IF YOU WERE PLANNING TO SUBMIT A PAPER FOR PRESENTATION AT APL 81, THE ACM-SPONSORED CONFERENCE IN SAN FRANCISCO, REMEMBER THE DEADLINE FOR SUBMISSION OF ABSTRACTS IS ONLY A FEW SHORT WEEKS AWAY. HOWEVER, THERE'S STILL TIME--AN ABSTRACT NEED ONLY BE A ONE- OR TWO-PARAGRAPH DESCRIPTION OF WHAT YOU EXPECT YOUR PAPER TO CONTAIN, SO YOU NEED NOT HAVE FULLY WORKED OUT YOUR IDEAS. THE FINAL PAPERS AREN'T DUE UNTIL FEBRUARY 1, 1981.

SUBMITTING AN ABSTRACT IS VERY EASY. YOU CAN TYPE )LOAD 1 APL81 ON THE SHARP SYSTEM AND YOU'LL GET A DESCRIPTION OF PROGRAMS THAT WILL ALLOW YOU TO ENTER, EDIT AND SEND THE TEXT OF YOUR ABSTRACT AUTOMATICALLY. IF YOU PREFER YOU CAN MAIL YOUR ABSTRACT TO THE PROGRAM CHAIRMAN AT THE FOLLOWING ADDRESS:

RICHARD J. ORGASS
XEROX CORPORATION
1350 JEFFERSON ROAD
ROCHESTER, NEW YORK 14623

(PLEASE NOTE THIS IS A NEWER ADDRESS THAN THAT GIVEN IN NEWS ITEM 148.)

FOR FURTHER INFORMATION ABOUT APL 81 SEE 1 NEWS, ITEM 148. WE LOOK FORWARD TO SEEING YOU AT THE CONFERENCE.

ITEM NO. 149 MAY 20, 1980
IMPROVED DISPLAY OF ERRORS OCCURRING IN <⎕TRAP> LINES

EFFECTIVE 24 MAY 1980, THE SHARP APL SYSTEM WILL IMPROVE THE WAY IN WHICH IT HANDLES THE DISPLAY OF ERRORS OCCURRING IN <⎕TRAP> RECOVERY EXPRESSIONS. THE IMPROVEMENTS WILL RESOLVE AMBIGUITIES IN THE DISPLAY OF THESE ERRORS, AND MAKE THE WAY IN WHICH THE SYSTEM TREATS <⎕TRAP> ERRORS EMANATING FROM LOCKED FUNCTIONS MORE CONSISTENT WITH THE WAY IT TREATS OTHER LOCKED FUNCTION ERRORS. BECAUSE THE MODIFICATIONS CHANGE THE DISPLAY OF NON-TRAPPABLE ERRORS ONLY, THEY WILL NOT AFFECT THE OPERATION OF EXISTING PROGRAMS.

APL CONTAINS FIVE BASIC MECHANISMS BY WHICH A LINE OF CODE CAN BE PUT INTO EXECUTION. THESE MECHANISMS ARE IMMEDIATE EXECUTION, FUNCTION CALL, ⍎, ⎕, AND <⎕TRAP>.

IN ORDER TO DISTINGUISH THE ENVIRONMENT IN WHICH AN ERROR OCCURRED, THE SYSTEM PREPENDS SOME STACK INFORMATION TO THE ERROR DISPLAY. IN THE CASE OF FUNCTION CALL, THE INFORMATION CONSISTS OF THE NAME AND LINE NUMBER OF THE FAILING FUNCTION; IN THE CASE OF IMMEDIATE EXECUTION AND <⎕TRAP>, IT CONSISTS OF THE ABSENCE OF A FUNCTION NAME AND LINE NUMBER. A QUAD LEVEL

IS REPRESENTED BY A PREFIX OF A SINGLE QUAD SYMBOL. ANY OF THESE THREE FORMS MAY BE FOLLOWED BY ONE OR MORE EXECUTE SYMBOLS, DENOTING EXECUTE LEVELS.

AT PRESENT, ERRORS OCCURRING IN IMMEDIATE EXECUTION AND ERRORS OCCURRING IN <⎕TRAP> RECOVERY EXPRESSIONS DISPLAY IN THE SAME FASHION, WITH NO ADDITIONAL INFORMATION INDICATING IN WHICH ENVIRONMENT THE ERROR OCCURRED.

TO RESOLVE THIS PROBLEM, SHARP APL WILL BEGIN TO PREFIX ERRORS OCCURRING IN <⎕TRAP> EXPRESSIONS BY THE SEQUENCE '⎕TRAP', RATHER THAN BY THE CURRENT SIX BLANKS. WITH THE NEW FORM OF DISPLAY, AN ERROR IN A <⎕TRAP> LINE MIGHT APPEAR AS FOLLOWS:

```
      0 DIV 0
SYNTAX ERROR
⎕TRAP 'CAN''T DIVIDE BY 0' ∧∨ Z←⍳0
                           ∧

      ⎕ER
   2 SYNTAX ERROR
⎕TRAP 'CAN''T DIVIDE BY 0' ∧∨ Z←⍳0
                           ∧
```

THE INDICATION THAT <⎕TRAP> WAS AT FAULT ALSO APPEARS IN <⎕ER>.

LIKE ERRORS OCCURRING IN DEFINED FUNCTIONS, ⎕, AND IMMEDIATE EXECUTION, ERRORS IN <⎕TRAP> EXPRESSIONS MAY INVOLVE EXECUTES. WHEN THEY DO, THE APPROPRIATE NUMBER OF EXECUTE SYMBOLS WILL APPEAR AFTER THE LETTERS '⎕TRAP' IN THE DISPLAY.

A SECOND CHANGE IN THE DISPLAY OF <⎕TRAP> ERRORS CONCERNS LOCKED FUNCTIONS. AT PRESENT, ERROR DISPLAY OF FAILING <⎕TRAP> LINES IS NOT COGNIZANT OF THE FACT THAT THE TRAP LINE MAY BE ASSOCIATED WITH A LOCKED FUNCTION. IT IS THEREFORE POSSIBLE TO HAVE THE <⎕TRAP> LINE IN ERROR DISPLAYED AT THE TERMINAL, EVEN THOUGH THE DISPLAY OF ERRORS EMANATING FROM EITHER EXECUTE OR THE FUNCTION ITSELF WOULD HAVE BEEN SUPPRESSED.

AFTER THE CHANGE, THE DISPLAY OF ERRORS OCCURRING IN <⎕TRAP> RECOVERY EXPRESSIONS WILL BE SUPPRESSED IF <u>BOTH</u> THE LEVEL AT WHICH THE TRAP EXPRESSION IS LOCALIZED <u>AND</u> THE LEVEL AT WHICH THE EXPRESSION IS EXECUTED ARE LOCKED. THESE TWO LEVELS ARE ALWAYS THE SAME FOR <⎕TRAP> EXPRESSIONS SPECIFYING THE CUTBACK ACTION CODE, <C>. HOWEVER, FOR <E>-TYPE EXPRESSIONS, THE LEVELS MAY BE DIFFERENT IF THE ERROR OCCURRED IN A FUNCTION OTHER THAN THE ONE IN WHICH <⎕TRAP> WAS LOCALIZED.

THE TABLE BELOW ILLUSTRATES THE DISPLAY POTENTIAL OF <⎕TRAP> LINES MORE CLEARLY. IN THE TABLE, '∇' REFERS TO A LEVEL OF THE STATE INDICATOR THAT CORRESPONDS TO AN UNLOCKED PROGRAM, WHILE '⍛' REFERS TO A LEVEL THAT CORRESPONDS TO A LOCKED ONE. ENTRIES MARKED WITH A 'Y' INDICATE THAT <⎕TRAP> ERRORS OCCURRING IN THE ASSOCIATED ENVIRONMENT <u>WILL</u> BE DISPLAYED; 'N' INDICATES THE OPPOSITE. ENTRIES MARKED WITH '-' DENOTE SITUATIONS THAT CAN'T ARISE, SINCE THE SAME LEVEL CAN'T BE BOTH LOCKED AND UNLOCKED.

| LEVEL OF <⎕TRAP> LOCALIZATION | ∇ | ⍛ | ∇ | ⍛ |
|---|---|---|---|---|
| LEVEL OF <⎕TRAP> EXECUTION | ∇ | ∇ | ⍛ | ⍛ |

| | | | | |
|---|---|---|---|---|
| <C>-TYPE EXPRESSION | Y | – | – | N |
| <E>-TYPE EXPRESSION | Y | Y | Y | N |

WHEN DISPLAY OF THE FAILING <⎕TRAP> EXPRESSION IS SUPPRESSED, THE ERROR WILL INSTEAD APPEAR AS FOLLOWS:

```
      ∇DIV∇

      0 DIV 0
SYNTAX ERROR
⎕TRAP ∇
      ^
```

ITEM NO. 148 APR 18, 1980
CALL FOR PAPERS FOR APL 81 CONFERENCE IN SAN FRANCISCO, OCTOBER 1981

APL 81, THE 12TH INTERNATIONAL APL USERS CONFERENCE, WILL BE HELD IN SAN FRANCISCO, CALIFORNIA, OCTOBER 21-23, 1981. APL 81 IS SPONSORED BY STAPL, AN INTERNATIONAL GROUP OF APL PROFESSIONALS AFFILIATED WITH THE ASSOCIATION FOR COMPUTING MACHINERY. TECHNICAL PAPERS ON ALL ASPECTS OF APL, INCLUDING THE FOLLOWING AREAS ARE SOLICITED.

- APL APPLICATIONS (ALL AREAS)
- APL--THE LANGUAGE
- APL IMPLEMENTATIONS (LARGE AND SMALL SYSTEMS)
- APL IN EDUCATION
- APL IN BUSINESS
- APL INTERFACES WITH OTHER SOFTWARE SYSTEMS
- APL SYSTEM ORGANIZATION AND MANAGEMENT
- APL COMPARED TO OTHER LANGUAGES

BOTH ABSTRACTS AND FULL PAPERS WILL BE REFEREED AND AUTHORS SHOULD SUBMIT THESE DOCUMENTS TO THE PROGRAM CHAIRMAN IN ACCORD WITH THE FOLLOWING SCHEDULE:

| | DUE DATE | AUTHOR NOTIFICATION |
|---|---|---|
| ABSTRACTS | OCTOBER 1, 1980 | DECEMBER 1, 1980 |
| FULL PAPERS | FEBRUARY 1, 1981 | APRIL 15, 1981 |

FULL COPIES OF COMPLETE PAPERS MUST BE RECEIVED BY THE PROGRAM CHAIRMAN BY JUNE 15, 1981 FOR INCLUSION IN THE CONFERENCE PROCEEDINGS AND FOR PRESENTATION AT THE CONFERENCE.

AUTHORS MAY SUBMIT ABSTRACTS USING THE FACILITIES CONTAINED IN THE SHARP WORKSPACE 1 APL81. INSTRUCTIONS FOR DOING SO ARE CONTAINED IN THAT WORKSPACE.

FURTHER INFORMATION MAY BE OBTAINED FROM EITHER OF THE FOLLOWING ADDRESSES:

PROGRAM CHAIRMAN
RICHARD J. ORGASS
DEPT. OF COMPUTER SCIENCE

CONFERENCE GENERAL CHAIRMAN
EUGENE R. MANNACIO
APL 81

VIRGINIA POLYTECHNIC INSTITUTE
P.O. BOX 17186
WASHINGTON, DC 20041

900 NORTH POINT STREET
SAN FRANCISCO, CA 94109

ITEM NO. 147 APR 03, 1980
REMOVAL OF SUPPORT FOR THE OLD VERSION OF THE FUNCTION <HSP>

WE HAVE REMOVED SUPPORT FOR AN OLD VERSION OF THE FUNCTION <HSP> IN THE WORKSPACE 1 HSPRINT: ATTEMPTS TO USE IT WILL RESULT IN A FILE ACCESS ERROR ON LINE [1] OF THE LOCKED FUNCTION <HSP>.

THIS WILL ONLY AFFECT APPLICATIONS WHICH WERE BUILT USING THIS OLD LOCKED VERSION: THE VERSION WHICH HAS BEEN RESIDENT IN WORKSPACE 1 HSPRINT FOR SOME TIME IS UNLOCKED AND WILL CONTINUE TO WORK. THE OLD VERSIONS, HOWEVER, SHOULD BE REPLACED WITH THE <HSP> FUNCTION COPIED FROM WORKSPACE 1 HSPRINT. NOTE THAT SOME OCCURRENCES OF THE OLD FUNCTION MAY BE IN PACKAGES ON FILES.

ITEM NO. 146 MAR 15, 1980
NEW FUNCTION <TERMINAL> SETS TERMINAL CHARACTERISTICS

A NEW MONADIC FUNCTION, CALLED <TERMINAL>, IS NOW AVAILABLE IN WORKSPACE 1 WSFNS . <TERMINAL> OPERATES IN A MANNER SIMILAR TO THE SYSTEM COMMAND )TERMINAL, AMD MAY BE USED TO PROGRAMMATICALLY SET THE TYPE OF TERMINAL IN USE.

THE SYNTAX OF <TERMINAL> IS:

```
      Z ← TERMINAL 'TYPE'
```

WHERE THE ARGUMENT 'TYPE' IS THE DESIRED NEW TERMINAL TYPE SETTING, AND THE RESULT <Z> IS A CHARACTER VECTOR IDENTIFYING THE PREVIOUS SETTING.

THE DEFAULT TERMINAL TYPE AT SIGN-ON IS 'IDLES'. TO SUPPRESS THE TRANSMISSION OF IDLE CHARACTERS AT THE END OF EACH LINE OF OUTPUT, USE:

```
      TERMINAL 'NOIDLES'
IDLES
```

AN EMPTY OR ALL-BLANK ARGUMENT RETURNS THE CURRENT TERMINAL SETTING WITHOUT CHANGE. FOR EXAMPLE:

```
      TERMINAL ''
NOIDLES

      ⍴TERMINAL ''
7
```

THE <TERMINAL> FUNCTION ACCEPTS THE SAME ARGUMENTS AS THE SYSTEM COMMAND, WITH THE EXCEPTION OF 'TEK4013' AND 'HP2641'. THESE TWO SETTINGS ARE NOT RECOGNIZED BY THE FUNCTION, AND WILL BE ACCEPTED BY THE SYSTEM COMMAND ONLY UNTIL 5 APRIL 1980.

IMPROPERLY FORMED ARGUMENTS TO <TERMINAL> CAUSE A 'RANK ERROR' OR A 'DOMAIN

ERROR' TO OCCUR. ARGUMENTS THAT ARE PROPERLY FORMED BUT DO NOT MATCH ANY VALID TERMINAL TYPE SETTING CAUSE AN 'INVALID TERMINAL TYPE' MESSAGE TO OCCUR.

THE <TERMINAL> FUNCTION IS EXPERIMENTAL, AND MAY HAVE ITS BEHAVIOUR MODIFIED IN THE FUTURE. FOR MORE INFORMATION ON THE SYSTEM COMMAND )TERMINAL AND THE TERMINAL TYPES THAT ARE SUPPORTED, PLEASE SEE NEWS ITEM 145 AND WORKSPACE 5 TERM .

ITEM NO. 145 MAR 15, 1980
CHANGES TO THE SYSTEM COMMAND )TERMINAL

THE SYSTEM COMMAND )TERMINAL HAS BEEN CHANGED TO SIMPIFY SELECTING WHETHER OR NOT THE SYSTEM INTRODUCES IDLES INTO TERMINAL OUTPUT.

WHEN YOU INITIALLY SIGN ON, THE SHARP APL SYSTEM TRANSMITS DATA IN A WAY SUITED TO SIMPLE HARDCOPY TERMINALS, TO AVOID ANY LOSS OF CHARACTERS. BECAUSE HARD COPY TERMINALS HAVE MOVING PARTS, THEY REQUIRE EXTRA TIME AT THE END OF EACH LINE OF OUTPUT, TO PERMIT THE PRINT MECHANISM TO RETURN TO THE LEFT MARGIN. SINCE THE SYSTEM TRANSMITS A STEADY, UNINTERRUPTED STREAM OF OUTPUT CHARACTERS, IT MUST INSERT A CALCULATED NUMBER OF IDLE CHARACTERS TO CAUSE THE PRINT MECHANISM TO CEASE PRINTING WHILE IT IS RETURNING TO THE LEFT MARGIN.

TERMINALS THAT DISPLAY DATA ON SCREENS RATHER THAN PAPER DO NOT NEED THESE IDLE CHARACTERS, BECAUSE THEY HAVE NO MOVING MECHANICAL PARTS. ALSO, MORE 'INTELLIGENT' HARDCOPY TERMINALS WITH BUILT-IN MEMORIES (OR BUFFERS) FREQUENTLY DON'T NEED IDLE CHARACTERS AT THE END OF EACH LINE.

THE CHANGES MADE TO )TERMINAL ARE AS FOLLOWS. PREVIOUSLY, THE DEFAULT TERMINAL SETTING WAS 'ASCII' FOR ASCII TERMINALS, AND 'IBM2741' FOR IBM 2741 OR EQUIVALENT TERMINALS. BOTH OF THESE INITIAL SETTINGS SUPPLIED IDLES AT THE END OF EACH OUTPUT LINE. NOW, THE DEFAULT FOR ALL TYPES OF TERMINALS IS 'IDLES'. TO SUPRESS THE IDLE CHARACTERS, ENTER )TERMINAL NOIDLES . THE SYSTEM RESPONDS BY DISPLAYING THE PREVIOUS STATE:

```
      )TERMINAL NOIDLES
WAS IDLES
```

THE TRANSMISSION OF IDLES CAN BE RESTORED BY TYPING:

```
      )TERMINAL IDLES
WAS NOIDLES
```

AS BEFORE, WHEN THE COMMAND IS USED WITHOUT AN ARGUMENT, THE CURRENT SETTING IS DISPLAYED:

```
      )TERMINAL
IS IDLES
```

THE SETTING )TERMINAL NOIDLES CAN OFFER A SIGNIFICANT INCREASE IN THE SPEED OF DATA DISPLAY, AS WELL AS A DECREASE IN THE NUMBER OF CHARACTERS TRANSMITTED BY THE SYSTEM. HOWEVER, WHEN USING THIS SETTING, A BUFFERED HARDCOPY TERMINAL MAY BEGIN TO LOSE CHARACTERS AND BEHAVE ERRATICALLY IF ITS INTERNAL BUFFER BECOMES COMPLETELY FILLED UP. THIS IS MOST LIKELY TO HAPPEN WHEN DISPLAYING APL CODE, WHICH TENDS TO HAVE A LOT OF UPPER AND

LOWER CASE SHIFTS IN IT. IN SUCH CASES, THE SETTING )TERMINAL NOIDLES SHOULD NOT BE USED.

THE SETTINGS 'TEK4013' AND 'HP2641' HAVE BEEN DIRECTLY SUPERSEDED BY THE 'NOIDLES' SETTING. THEREFORE, EFFECTIVE 5 APRIL 1980, ARGUMENTS OF 'TEK4013' AND 'HP2641' WILL NO LONGER BE ACCEPTED. OTHER NON-GENERIC TERMINAL TYPES, SUCH AS 'GET1200', 'SDATA', 'TY33', AND SO ON WILL CONTINUE TO BE VALID. AN UP-TO-DATE LIST OF WHAT TERMINAL TYPES ARE SUPPORTED CAN BE FOUND BY TYPING <TERMINALS> IN WORKSPACE 5 TERM .

LIKE OTHER SYSTEM COMMANDS, THE )TERMINAL COMMAND MAY BE ABBREVIATED TO FOUR LETTERS, VIZ )TERM .

ITEM NO. 144 FEB 08, 1980
'COMPRESSION' EXTENDED TO 'REPLICATION'

THE SHARP APL SYSTEM NOW PROVIDES AN EXTENDED VERSION OF THE / OPERATOR. ALL EXISTING USES OF COMPRESSION WORK JUST AS THEY USED TO, BUT SOME NEW CAPABILITIES HAVE BEEN ADDED. THE EXTENDED VERSION IS CALLED REPLICATION.

FORMERLY THE VALUES IN THE ARRAY USED AS THE LEFT ARGUMENT OF / WERE REQUIRED TO BE EITHER 1 OR 0. THE DERIVED COMPRESSION FUNCTION SELECTED FROM ITS RIGHT ARGUMENT EITHER ONE OR ZERO REPLICAS OF EACH SUBARRAY WITHIN THE RIGHT ARGUMENT.

THE CHANGE FROM COMPRESSION TO REPLICATION MEANS THAT THE LEFT ARGUMENT OF / IS NO LONGER RESTRICTED TO THE VALUES 1 OR 0. INSTEAD, EACH ELEMENT MAY BE ANY NON-NEGATIVE INTEGER. REPLICATION WORKS IN A MANNER SIMILAR TO COMPRESSION. FOR EACH ELEMENT OF THE LEFT ARGUMENT, THE RESULT CONTAINS THAT MANY REPLICAS OF THE CORRESPONDING SUBARRAY OF THE RIGHT.

EXAMPLE 1

```
      X←3 10⍴'  A ROSE J JOPLIN  IS A ROSE 'S
      X
  A ROSE
J JOPLIN
IS A ROSE

      ,1 0 2/[⎕IO] X
  A ROSE IS A ROSE IS A ROSE
```

EXAMPLE 2

```
      ,3≠1 3⍴'HO '
HO HO HO
```

EXAMPLE 3

ADDING A SINGLE COLUMN OF NUMBERS (C) TO EACH COLUMN OF A MATRIX (M) MAY NOW BE DONE AS FOLLOWS:

```
      M+(¯1↑⍴M)/C
```

ITEM NO. 143 FEB 07, 1980

## CHANGES TO THE BEHAVIOUR OF BRANCH AND ⎕LC

EFFECTIVE 3 MAY 1980, THE SHARP APL SYSTEM WILL REVISE ITS TREATMENT OF CERTAIN CLASSES OF BRANCH, AND WILL ALSO REVISE THE WAY IT REPRESENTS EXECUTE (⍎) AND QUAD (⎕) IN THE LINE COUNTER, ⎕LC . THESE CHANGES WILL HELP TO ACHIEVE THE FOLLOWING GOALS:

1. THE DEFINITION OF BRANCH WILL BE SIMPLE AND CONSISTENT. THE BEHAVIOUR OF BRANCH WILL NO LONGER DEPEND ON WHETHER IT WAS INVOKED FROM A FUNCTION, FROM IMMEDIATE EXECUTION, FROM ⎕TRAP, OR FROM ⍎ . REGARDLESS OF CONTEXT, BRANCH TO ZERO AND BRANCH TO AN EMPTY VECTOR WILL ALWAYS HAVE THE SAME MEANING. IN PARTICULAR, AFTER THE CHANGES, EACH OF THE STATEMENTS →X, ⍎'→X', AND →⍎'X' WILL BEHAVE IDENTICALLY (WHEREAS AT PRESENT THEY DON'T ALWAYS).

2. BRANCH FROM WITHIN A STATEMENT EXECUTED VIA ⍎ (WHICH IS HOW THE LATENT EXPRESSION ⎕LX IS EXECUTED) WILL BEHAVE IN A WAY THAT IS COMPATIBLE WITH <u>BOTH</u> THE CURRENT SHARP APL SYSTEM <u>AND</u> IBM APL SYSTEMS.

## <u>CHANGES AFFECTING TRANSFER OF CONTROL</u>

THE LIST BELOW SUMMARIZES THE CHANGES AFFECTING BRANCH OPERATIONS:

1. ALL BRANCHES TYPED FROM WITHIN ⎕ MODE WILL RESULT IN A 'SYNTAX ERROR'. (AT PRESENT, →0 AND →⍳0 ARE PERMITTED, AND ⎕ RETURNS THE VALUE 0 OR ⍳0 RESPECTIVELY, BUT NO OTHER BRANCH IS PERMITTED.)

2. ALL BRANCHES OF THE FORM R←⍎'→X' WILL RESULT IN A 'SYNTAX ERROR'. (AT PRESENT, THIS IS PERMITTED WHEN <X> IS 0 OR ⍳0 , AND <R> IS ASSIGNED THE VALUE 0 OR ⍳0 RESPECTIVELY, BUT NO OTHER VALUE OF <X> IS PERMITTED.)

3. AN IMMEDIATE-EXECUTION LINE OR ⎕TRAP RECOVERY EXPRESSION ENDING WITH →⍳0 WILL BEHAVE AS IF THE BRANCH WERE NOT THERE. (AT PRESENT, SUCH A BRANCH CAUSES THE FUNCTION HIGHEST ON THE STATE INDICATOR TO RESUME AT ITS NEXT LINE, ⎕LC+1 .)

4. →0 FROM WITHIN EXECUTE WILL TERMINATE THE FUNCTION HIGHEST ON THE STATE INDICATOR. (AT PRESENT, ⍎'→0' TERMINATES EXECUTION OF THE ARGUMENT OF ⍎, BUT HAS NO EFFECT ON THE ENCOMPASSING FUNCTION.)

## <u>CHANGES AFFECTING THE LINE COUNTER</u>

THE CONTENTS OF THE LINE COUNTER, ⎕LC, WILL CHANGE SLIGHTLY FOR ELEMENTS THAT REPRESENT ⍎ OR ⎕ . IN SHARP APL (AS IN VS APL, BUT NOT AS IN APLSV), EACH USE OF ⍎ AND ⎕ HAS BEEN REPRESENTED BY ITS OWN ROW IN THE STATE INDICATOR. FURTHER, IN SHARP APL EACH USE OF ⍎ OR ⎕ HAS ITS OWN ELEMENT IN ⎕LC; EACH SUCH ELEMENT HAS HAD THE VALUE ZERO. (BY CONTRAST, IN BOTH VS APL AND APLSV, ⎕LC CONTAINS <u>NO</u> ELEMENTS CORRESPONDING TO ⍎ OR ⎕; HENCE, IN VS APL, ⎕LC MAY HAVE FEWER ELEMENTS THAN THERE ARE ROWS IN THE STATE INDICATOR.)

AFTER THE CHANGE, THE STATE INDICATOR WILL REMAIN AS IT IS, AND ⎕LC WILL

CONTINUE TO HAVE AN ELEMENT CORRESPONDING TO EACH LEVEL OF THE STATE INDICATOR. HOWEVER, THE VALUE OF EACH ELEMENT OF ⎕LC THAT CORRESPONDS TO A USE OF ⍎ OR ⎕ WILL BE THE LINE NUMBER OF THE DEFINED FUNCTION THAT INVOKED IT, RATHER THAN ZERO. WHEN ⍎ OR ⎕ IS USED IN IMMEDIATE EXECUTION MODE WITH A SUSPENDED FUNCTION ON THE STATE INDICATOR, ⎕LC WILL CONTAIN THE LINE NUMBER ON WHICH THAT FUNCTION WAS SUSPENDED. ONLY WHEN ⍎ OR ⎕ IS USED IN IMMEDIATE EXECUTION MODE WITH NO DEFINED FUNCTION ON THE STATE INDICATOR WILL ⎕LC CONTAIN A 0 .

THIS IS THE FIRST OF TWO NEWS ITEMS RELATING TO THE CHANGES TO BRANCH AND ⎕LC . FOR MORE INFORMATION REGARDING THE IMPACT OF THE MODIFICATIONS ON EXISTING PROGRAMS, PLEASE SEE ITEM 142.

ITEM NO. 142 FEB 07, 1980
IMPACT OF THE CHANGES TO BRANCH AND ⎕LC

THIS IS THE SECOND OF TWO NEWS ITEMS RELATING TO THE CHANGES TO BRANCH AND ⎕LC . FOR DETAILS OF THE MODIFICATIONS, AND THE REASONS BEHIND THEM, PLEASE SEE ITEM 143.

THE PRIMARY EFFECT OF THE CHANGE TO ⎕LC IS AS FOLLOWS. TO PROVIDE AUTOMATIC RESTARTING OF AN INTERRUPTED TASK, THE LATENT EXPRESSION ⎕LX MUST CONTAIN A BRANCH. HOWEVER, WHEN THE SYSTEM EXECUTES ⎕LX, IT DOES SO (NOW AND AFTER THE CHANGE) BY INVOKING ⍎, AND THEREFORE ⍎ GOES ON THE STATE INDICATOR, AND ⎕LC'S FIRST ELEMENT CORRESPONDS TO THAT ⍎ . BECAUSE THE FIRST ELEMENT OF ⎕LC DESCRIBES THE SYSTEM'S USE OF ⍎ (RATHER THAN THE FUNCTION SUSPENDED IN THE WORKSPACE), IN SHARP APL AS IT IS AT PRESENT, A LATENT EXPRESSION TO RESUME EXECUTION WHERE WORK WAS INTERRUPTED MUST USE →1↓⎕LC RATHER THAN →⎕LC . THE 1↓ IS THERE TO DISPOSE OF THE LEADING ZERO IN THE LINE COUNTER. BY CONTRAST, PROGRAMS RUNNING IN IBM APL SYSTEMS MUST USE →⎕LC INSTEAD.

AFTER THE CHANGE, →⎕LC AND →1↓⎕LC, WHEN THEY OCCUR IN THE LATENT EXPRESSION OR IN ANY OTHER ARGUMENT TO ⍎, WILL HAVE THE SAME EFFECT. LATENT EXPRESSIONS PREPARED FOR THE PRESENT SHARP SYSTEM AND THOSE PREPARED FOR IBM SYSTEMS WILL BOTH RUN CORRECTLY. THE CHANGE WILL NOT AFFECT THE OPERATION OF EXISTING PROGRAMS RUNNING ON SHARP APL, UNLESS A PROGRAM EXPLICITLY ASSUMES THAT, WITHIN ⍎ OR ⎕, 1↑⎕LC IS 0 .

MOST USES OF BRANCH (WHICH CORRESPOND TO SIMPLE BRANCHES TO LINES WITHIN A FUNCTION) WILL NOT BE AFFECTED AT ALL. BRANCHING TO A NONZERO INTEGER (SUCH AS A LINE LABEL) IN A FUNCTION, ⍎, ⎕TRAP, OR IMMEDIATE EXECUTION, WILL CONTINUE TO BEHAVE EXACTLY AS IT DOES AT PRESENT.

OF THE FOUR CATEGORIES OF BRANCHES WHOSE OPERATION WILL BE AFFECTED, THE FIRST TWO CATEGORIES (BRANCHES WITHIN ⎕ MODE, AND BRANCHES WITHIN EXECUTE WHERE A RESULT IS EXPECTED) ARE ESOTERIC CASES WHICH ARE EXCEEDINGLY UNLIKELY TO OCCUR.

BRANCHES BELONGING TO THE THIRD CATEGORY (→⍳0 IN IMMEDIATE EXECUTION OR IN ⎕TRAP) ARE ALSO UNLIKELY TO OCCUR. ⎕TRAP EXPRESSIONS WHICH MAKE USE OF THE PRESENT BEHAVIOUR OF →⍳0 SHOULD BE CHANGED TO PERFORM →⎕LC+1 INSTEAD.

THE FINAL CATEGORY OF BRANCH CHANGES (→0 WITHIN EXECUTE) IS THE ONE MOST LIKELY TO AFFECT THE BEHAVIOUR OF EXISTING PROGRAMS. SEVERAL SIMPLE TECHNIQUES THAT ARE UNALTERED BY THE UPCOMING CHANGES MAY BE USED TO

CONVERT EXISTING STATEMENTS. THE STATEMENTS SUGGESTED HERE WILL WORK BOTH BEFORE THE CHANGE AND AFTER IT.

AS AN EXAMPLE, SUPPOSE THE FOLLOWING LINE OF CODE IS USED IN A PROGRAM:

```
[4]   CN←CN+1 ◊ ⍎PRFLG/'J←0 ◊ →(×100|CN)⍴0 ◊ CN' ◊ K←1↓K
```

THE LINE FIRST ADDS 1 TO <CN> EACH TIME THROUGH; THEN, IF <PRFLG> IS 1, IT EXECUTES AN EXPRESSION WHICH SETS <J> TO 0, AND PRINTS THE VALUE OF <CN> IF IT IS A MULTIPLE OF 100 . THE LAST STATEMENT ON THE LINE, WHICH IS ALWAYS EXECUTED, DISCARDS THE FIRST ELEMENT OF <K>.

AN EQUIVALENT LINE WHICH USES NESTED EXECUTES TO ACHIEVE THE SAME EFFECT THAT THE COMPUTED →0 ABOVE DOES, CAN BE CONSTRUCTED AS FOLLOWS:

```
[4]   CN←CN+1 ◊ ⍎PRFLG/'J←0 ◊ ⍎(0=100|CN)/''CN''' ◊ K←1↓K
```

AN ALTERNATE METHOD MAKES USE OF SEPARATE LINES AND AN ADDITIONAL LINE LABEL, WITH THE SAME EFFECT:

```
[4]   CN←CN+1 ◊ ⍎PRFLG/'J←0 ◊ →(×100|CN)⍴L0 ◊ CN'
[5]  L0:K←1↓K
```

STILL ANOTHER METHOD INVOLVES REMOVING THE USE OF EXECUTE ALTOGETHER -- OFTEN THE BEST CHOICE, SINCE EXECUTE IS FREQUENTLY OVERUSED IN CASES WHERE STRAIGHTFORWARD CONDITIONAL BRANCH LOGIC WOULD BE BOTH EASIER TO READ AND MORE EFFICIENT:

```
[4]   CN←CN+1 ◊ →PRFLG↓L0 ◊ J←0 ◊ →(×100|CN)⍴L0 ◊ CN
[5]  L0:K←1↓K
```

EVEN SIMPLER CODING TRANSFORMATIONS MAY EXIST, DEPENDING UPON THE EXACT NATURE OF THE LINE. FOR EXAMPLE, IF THE ORIGINAL LINE ABOVE DID NOT CONTAIN THE ASSIGNMENT OF <J> TO 0, THEN AN EASIER CONVERSION THAT INVOLVES THE SAME NUMBER OF EXECUTES AND NO NEW LINES WOULD CHANGE IT TO:

```
[4]   CN←CN+1 ◊ ⍎(PRFLG>100|CN)/'CN' ◊ K←1↓K
```

A SIMPLIFICATION OF THE NEW BEHAVIOUR OF ⍎'→0' IS THAT IT NOW BECOMES STRAIGHTFORWARD TO CAUSE THE FUNCTION INVOKING AN EXECUTE TO TERMINATE FROM WITHIN THE EXECUTE. FOR EXAMPLE:

```
[3]   ⍎(2<⍴⍴X)/'⎕UNTIE TN ◊ →0'
```

WITH THE SYSTEM'S CURRENT BEHAVIOUR, THE SELF-EXPLANATORY →0 DOES NOT HAVE THE DESIRED EFFECT, AND YOU MUST INSTEAD REPLACE IT BY A BRANCH TO ¯1 OR SOME OTHER NONZERO VALUE FOR WHICH THERE IS NO LINE IN THE FUNCTION.

USERS WHO MAKE USE OF ANY OF THE BRANCH FACILITIES THAT WILL CHANGE, NOTABLY →0 IN EXECUTE, SHOULD ATTEMPT TO LOCATE AND ALTER ALL OCCURRENCES OF THEM AS SOON AS POSSIBLE. PLEASE DON'T LEAVE THIS UNTIL THE LAST MINUTE, AS YOU MAY FORGET AND FIND THAT, AFTER 3 MAY 1980, YOUR PROGRAMS NO LONGER FUNCTION AS YOU HAD ANTICIPATED. THE STAFF OF I. P. SHARP ASSOCIATES WILL BE PLEASED TO ANSWER ANY SPECIFIC QUESTIONS YOU MAY HAVE, AND TO ASSIST YOU IN THE CONVERSION TASK.

ITEM NO. 141 JAN 17, 1980

2882106 DATE 11/30/80, CLOCK 23.46.50

CHANGE TO )LIB - EFFECTIVE SATURDAY 19 JANUARY.

)LIB WILL DISPLAY WORKSPACE NAMES IN ALPHABETIC ORDER.

ITEM NO. 140 JAN 16, 1980
DR. KENNETH E. IVERSON JOINS I.P. SHARP ASSOCIATES

DR. KENNETH E. IVERSON HAS LEFT IBM AND HAS JOINED I.P. SHARP ASSOCIATES. HE WILL BE MOVING TO TORONTO TO WORK WITH THE SYSTEM DEVELOPMENT GROUP PRIMARILY ON EXTENSIONS TO THE APL LANGUAGE. HE ALSO EXPECTS TO MAINTAIN HIS ACTIVE INTEREST IN THE USE OF APL IN TEACHING.

KEN IS A NATIVE OF ALBERTA AND A GRADUATE OF QUEEN'S UNIVERSITY IN KINGSTON. FROM THERE HE WENT TO HARVARD AND STUDIED FOR HIS DOCTORATE UNDER PROFESSOR HOWARD AIKEN. IT WAS AT HARVARD THAT APL WAS CONCEIVED AND IT WAS SUBSEQUENTLY DOCUMENTED IN THE BOOK '*A PROGRAMMING LANGUAGE*' PUBLISHED IN 1962, SOME TWO YEARS AFTER HE JOINED IBM.

DR. IVERSON WAS WITH IBM FOR NEARLY TWENTY YEARS AND WAS A FELLOW AT THE T.J. WATSON RESEARCH CENTER. IN THOSE YEARS HE HAS SEEN THE LANGUAGE GROW FROM AN IDEA TO A FULL-SCALE COMMERCIAL PROGRAMMING LANGUAGE. HE WAS RECENTLY HONOURED BY THE ACM TURING AWARD AND ELECTION TO THE NATIONAL ACADEMY OF ENGINEERING.

WE ALL WELCOME KEN TO I.P. SHARP ASSOCIATES AND LOOK FORWARD TO WORKING WITH HIM.